VPI-E-91-09

A Study of the Response of Nonlinear Springs

Final Report

M. W. Hyer[1], Principal Investigator
T. W. Knott[2] and E. R. Johnson[3], Co-investigators

Performance period: August 25, 1988 - August 15, 1990

NASA Co-operative Agreement NCC-1-130

Virginia Polytechnic Institute and State University
Blacksburg, VA 24061-0219

July 1991

Technical Monitor: Dr. Gary Farley
Materials Division, Applied Materials Branch
National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia 23665-5225

---

1 Professor, Department of Engineering Science and Mechanics,

2 Research Associate, Department of Engineering Science and Mechanics,

3 Associate Professor, Department of Aerospace and Ocean Engineering

ABSTRACT

The various phases to developing a methodology for studying the response of a spring-reinforced arch subjected to a point load are discussed. The arch is simply supported at its ends with both the spring and the point load assumed to be at midspan. The spring is present to off-set the typical snap-through behavior normally associated with arches, and to provide a structure that responds with constant resistance over a finite displacement. The various phases discussed consist of: Development of the closed-form solution for the shallow arch case; Development of a finite-difference analysis to study (shallow) arches; and; Development of a finite-element analysis for studying more general shallow and nonshallow arches. The two numerical analyses rely on a continuation scheme to move the solution past limit points, and to move onto bifurcated paths, both characteristics being common to the arch problem. An eigenvalue method is used for a continuation scheme. The finite-difference analysis is based on a mixed formulation (force and displacement variables) of the governing equations. The governing equations for the mixed formulation are in first order form, making the finite-difference implementation convenient. However, as will be discussed, the mixed formulation is not well-suited for the eigenvalue continuation scheme. This provided the motivation for the displacement-based finite-element analysis. Both the finite-difference and the finite-element analyses are compared with the closed-form shallow arch solution. Agreement is excellent, except for the potential problems with the finite-difference analysis and the continuation scheme. Agreement between the finite-element analysis and another investigator's numerical analysis for deep arches is also good.

INTRODUCTION

Background

In the initial phase of this study, closed-form solutions for the response of a shallow arch reinforced with a single spring at midspan and loaded with a downward point load, also at midspan, were developed. Typical results were documented in ref. [1]. A schematic of this physical situation is shown in fig. 1, a figure taken from ref. [1]. The purpose of reinforcing the arch with a spring is to alter the arch's well-known snap-through behavior associated with the limit point. With no spring reinforcement, under load control the load-midspan deflection behavior is characterized by a sudden and large increase in deflection as the load reaches a limit value. Snap-through behavior is characterized in fig. 2a, the load increasing to limit point L, suddenly 'jumping' (dynamically) to point M, and then continuing on to point N. With a reinforcing spring, the snap-through behavior can be controlled. The load-midspan deflection relation that can result from the addition of a reinforcing spring is shown in fig. 2b. Instead of the load increasing to the limit point, the load increases to point C, then the relation proceeds along path CD (which is stabilized by the center sping) to point D, and then on to point N. In a particular application, to take advantage of this altered response it may be desirable to have a steeper or a shallower slope to path CD. The slope of path CD can be controlled by the stiffness of the spring. Relationships between the spring stiffness, the geometric and elastic properties of the arch, and the characteristic of arch response were summarized in fig. 11 in ref. 1.

Since a generalization of the reinforcing concept would consider arches that cannot be categorized as shallow, efforts then focused on extending the analysis to include deep arches. Unfortunately, closed-form solutions to the deep arch problem, even with no spring, do not exist. Thus the extension of the spring reinforcement concept to these other geometries relied on a numerical method. With a numerical approach, the problems of zero stiffness at limit points, such as point L in fig. 2a, and bifurcated

(multivalued) solution points, such as point C in fig. 2b, are serious. Most standard numerical schemes become singular at these points. Fortunately, there are techniques to overcome the singular nature of the problem at these points, though they are not in the category of general purpose techniques. To study deep arches, then, the following steps were necessary:

1) A numerical approach had to be developed and verified; and

2) A technique to overcome the singular nature of the problem had to be implemented.

During the second phase of this study, both steps were accomplished using a finite-difference approach. Specifically, a finite-difference approach was used with the incremental equations for this geometrically nonlinear problem. The incremental equations were, of course, linear and to solve the problem the linear equations were solved repeatedly as the load level was increased. To check the formulation of the finite-difference approach, the shallow arch problem was resolved using the finite-difference scheme. The numerical solution was compared with limited closed-form results in ref. [2]. The comparison between the closed-form and finite-difference approaches was excellent. A continuation of the work with the finite-difference approach focused on the implementation of a technique to overcome singularities, and on a further comparison of the numerical results with the closed-form solution, particularly the ability to move the solution through limit points, and to move the solution onto one branch or the other at bifurcation points. The results of this phase of the work have not been reported on and will be discussed in a subsequent section.

As will be seen, the finite-difference method, including the method to overcome singularities, worked quite well...on the problems tested. The finite-difference formulation was based on the first-order form of the incremental equations governing the behavior of a shallow arch. These incremental equations, by their first-order nature, were of a mixed formulation, i.e., the equations involved both force and displacement variables. While the first-order mixed form of the equations was very

convenient for implementing the finite-difference approach, the mixed feature proved to be a serious drawback. The reason is as follows: The finite-difference formulation of the linear incremental equations was written as a known coefficient matrix times the unknown increments in the force and displacement variables equal to a known vector. At each load level the unknown increments were determined from the set of linear equations using standard methods. At limit or bifurcation points, the coefficient matrix was singular and the equations could not be solved. To know when singular points were to be encountered, the eigenvalues of the coefficient matrix were computed, one eigenvalue going to zero when the matrix became singular. So the solution could proceed past the singular points, the eigenvector associated with the singular eigenvalue, but evaluated at a load slightly away from the singularity, was used to represent the unknown increments at the singular points. Using the eigenvector representation of the increments, the solution could be made to proceed through the singularity. Once through the singularity, the increments were again found by using the coefficient matrix, the matrix being non-singular again. Unfortunately, with a mixed formulation, the coefficient matrix is not symmetric so the eigenvalues and vectors are in general complex, thus it was difficult to use them reliably to predict when singular behavior was about to occur. In addition, with complex eigenvectors, a physical interpretation of what was taking place in the vicinity of the singular points was difficult. As an alternative to predicting when a singular point was being approached, a dynamic stability analysis was posed. At each level of applied load, the frequency of small motions about the static equilibrium configuration can be used to study stability. Bifurcation and limit points are associated with the lowest frequency going to zero. Keeping track of the frequencies computed from the stability analysis would thus provide insight into the location of singular points. This dynamic stability method would also have the advantage of indicating the stability characteristics of the various paths associated with the singular points. Unfortunately, with the mixed formulation, the mass matrix for the problem was itself singu-

lar. The mass matrix was not a mass matrix in the classic sense of a purely displacement based formulation. As all variables were not displacements and hence the masses associated with the nondisplacement variables were not really masses. The eigenvalues and eigenvectors of the dynamic system were also complex and did not provide any benefit to the analysis of limit and bifurcation point. Thus, because of the difficulties with the mixed approach, efforts focused on the development of a displacement-based finite-element formulation.

A displacement-based formulation would result in a symmetric tangent stiffness matrix which has real eigenvalues, and hence there is less difficulty in determining the stability of the equilibrium paths. The eigenvectors would also be real and hence more useful. This displacement formulation was more general than the finite-difference formulation in that it was not restricted to shallow arches. A subsequent section of this report will trace the development of that work.

Overview of Report

To follow in this report, then, are a number of sections. The next section presents the equations governing the behavior of a shallow arch with a reinforcing spring. The case of no supporting spring is a special case of these equations. The third section summarizes the finite-difference representation of incremental form of the shallow arch equations. In the fourth section results from the finite-difference approach are compared with closed-form results, particularly for branched solutions. This section will demonstrate the ability to move through singular points. In the fifth section the finite-element formulation of the problem is presented. In the sixth section, results obtained by the finite-element method for several shallow arches are compared with the closed-form solution. Then the finite-element results for deep arches are compared with the numerical results of Huddleston [3]. Huddleston obtained results by using the so-called shooting method for solving the first-order form of the equations for the deep arch.

Three appendices are also included. Appendix A is a detailed derivation of the governing equations for both deep and shallow arches. Appendix B is a users guide to the finite-element program used to obtain numerical results. Appendix C is a listing of the finite-element program.

EQUATIONS GOVERNING THE BEHAVIOR OF A SHALLOW ARCH

The governing equations are derived using the first variation of the total potential energy of the arch-spring-load system shown in Fig. 1. The initial shape of the arch is described by the function $z_o(x)$, $z_o(x)$ representing an initial shape relative to a straight line connecting the arch supports. Because the arch is shallow, there is no distinction between the arc length coordinate along the arch, s_o, and the Cartesian coordinate x measured along the straight line between the supports. Here arch midspan is designated as $x = 0$. With this, the total potential energy, V, is given by

$$V = \frac{1}{2}\int_{\frac{-L}{2}}^{\frac{L}{2}} \left\{EA\left[u' + z'_o w' + \frac{1}{2}(w')^2\right]^2 + EI(-w'')^2\right\} dx + Pw(0) + \frac{1}{2}Kw^2(0), \tag{1}$$

where u is the displacement in the x direction (horizontal), w the displacement in the z-direction (vertical), E the Young's modulus of the arch, A and I the area and the area moment of inertia at the arch cross-section, P the applied load, and K the spring stiffness. (The development of eq. (1) is given in Appendix A.)

Equilibrium Equations and Boundary Conditions

Taking the first variation of the total potential energy and defining the force and moment resultants to be

$$N = EA\left[u' + z'_o w' + \frac{1}{2}(w')^2\right]$$
$$M = EI(-w'') \tag{2}$$

results in

$$\delta V = \int_{\frac{-L}{2}}^{\frac{L}{2}} \{N[\delta u' + z'_o \delta w' + w' \delta w'] + M(-\delta w'')\} dx + P\delta w(0) + Kw(0)\delta w(0) . \tag{3}$$

Integrating by parts twice yields

$$\delta V = \int_{\frac{-L}{2}}^{\frac{L}{2}} \{-N'\delta u - [N(z'_o + w'_o)]'\delta w - M''\delta w\} dx + N\delta u \Big|_{\frac{-L}{2}}^{\frac{L}{2}} + (M' + N(z'_o + w'))\delta w \Big|_{\frac{-L}{2}}^{0^-} \Big|_{0^+}^{\frac{L}{2}} - M\delta w' \Big|_{\frac{-L}{2}}^{0^-} \Big|_{0^+}^{\frac{L}{2}} + [P + Kw(0)]\delta w(0) , \tag{4}$$

where ( )' denotes differentiation with respect to x. Setting $\delta V = 0$ gives the equilibrium equations and boundary terms. The equilibrium equations are

from δu: $\quad N' = 0$ (5)

from δw: $\quad M'' + [N(z'_o + w')]' = 0 \quad x \in (-\frac{L}{2}, 0^-)$ and $x \in (0^+, \frac{L}{2})$, (6)

and the boundary terms are

$$N\delta u \Big|_{\frac{-L}{2}}^{\frac{L}{2}} + [M' + N(z'_o + w')]\delta w \Big|_{\frac{-L}{2}}^{\frac{L}{2}} - (M' + N(z'_o + w'))\delta w \Big|_{x=0^+} + (M' + N(z'_o + w'))\delta w \Big|_{x=0^-} + (P + Kw)\delta w \Big|_{x=0} - M\delta w' \Big|_{\frac{-L}{2}}^{\frac{L}{2}} + M\delta w' \Big|_{x=0^+} - M\delta w' \Big|_{x=0^-} . \tag{7}$$

From the boundary terms the boundary conditions are

$$\text{at} \quad x = \pm \frac{L}{2}, \qquad u = 0, \quad w = 0, \quad M = 0. \tag{8}$$

The conditions at the location of the load and spring can be written as

$$\text{at} \quad x = 0 \qquad w(0^-) = w(0^+) \tag{8}$$

$$w'(0^-) = w'(0^+) \tag{9}$$

$$M(0^-) = M(0^+) \tag{10}$$

and

$$V(0^-) + P + Kw(0) = V(0^+), \tag{11}$$

where the shear V is

$$V = M' + (z_o + w)'N. \tag{12}$$

Equations 8-10 are referred to as the continuity conditions while eq. 11 is referred to as the jump condition. Due to eq. 5, the thrust, N, is constant, i.e.,

$$N = \text{constant}. \tag{13}$$

Thus the boundary value problem becomes

$$(-EIw'')'' + N(z''_o + w'') = 0 \qquad x \in (\frac{-L}{2}, 0^-) \text{ and } x \in (0^+, \frac{L}{2}), \tag{14}$$

with

$$N = \frac{EA}{L} \int_{\frac{-L}{2}}^{\frac{L}{2}} \left[z'_o w' + \frac{1}{2}(w')^2 \right] dx \tag{15}$$

and boundary and continuity conditions, given by eqs. 8-11. Nondimensionalizing by defining

$$\bar{x} = \frac{2x}{L}, \qquad \bar{z}_o = \frac{z_o}{2}\sqrt{\frac{A}{I}} = \lambda\left(1 - \bar{x}^2\right)$$
$$\bar{w} = \frac{-w}{2}\sqrt{\frac{A}{I}}, \quad \gamma^2 = \frac{-NL^2}{4EI}, \quad \lambda = \frac{\sqrt{3}\,H}{h} \tag{16}$$
$$k = \frac{KL^3}{8EI}, \quad p = \frac{PL^3}{16EI}\sqrt{\frac{A}{I}},$$

yields the governing equation

$$\bar{w}'''' - \gamma^2[\bar{z}''_o - \bar{w}''] = 0, \tag{17}$$

with the nondimensional version of eq. 15,

$$\gamma^2 = \int_{-1}^{1}\left(2\bar{z}'_o\bar{w}' - (\bar{w}')^2\right)d\bar{x}, \tag{18}$$

and the boundary conditions

$$\bar{w}(-1) = 0, \quad \bar{w}(1) = 0$$
$$\bar{w}''(-1) = 0, \quad \bar{w}''(1) = 0, \tag{19}$$

and continuity and jump conditions

$$\bar{w}(0^-) = \bar{w}(0^+)$$
$$\bar{w}'(0^-) = \bar{w}'(0^+)$$
$$\bar{w}''(0^-) = \bar{w}''(0^+) \tag{20}$$
$$\bar{w}'''\Big|_{0^-} + p - k\bar{w} = w'''\Big|_{0^+}.$$

The quantity γ^2 now represents the thrust. For a circular arch, function z_o is given by eq. A.70 in Appendix A and in nondimensional form in eq. 16. The solution to eq. 17 for the circular arch is

$$\bar{w} = \begin{cases} A_1 \sin \gamma\bar{x} + A_2 \cos \gamma\bar{x} + A_3\bar{x} + A_4 - 2\lambda \dfrac{\bar{x}^2}{2}, & x \in (-1,0) \\ A_5 \sin \gamma\bar{x} + A_6 \cos \gamma\bar{x} + A_7\bar{x} + A_8 - 2\lambda \dfrac{\bar{x}^2}{2}, & x \in (0,1) \end{cases} \tag{21}$$

which, with the boundary and jump conditions, leads to the matrix equation,

$$C_{ij}A_i = \lambda c_i + p d_i \; ; \; i,j = 1,2,\ldots,8 \tag{22}$$

for the eight unknown A_i's. Equation 18 results in

$$m_{ij} A_i A_j + n_i A_i + q = 0 . \tag{23}$$

This leads to a quadratic equation in the nondimensional load p such that for given values of γ, λ, and k, the nondimensional load p can be determined.

The Adjoint Problem

The solution for $\bar{w}$, eq. 21, is unique if γ is not an eigenvalue of the adjoint problem to the differential equation, eq. 17, boundary conditions, eq. 19, and the transition conditions, eq. 20. If γ is an eigenvalue to the adjoint problem, then multiple solutions for $\bar{w}$ are possible for particular values of the load p, and matrix C_{ij} in eq. 22 is singular. (The fact that multiple solutions to eq. 17 and its associated boundary and transition conditions are possible is sometimes referred to as Fredholm's alternative theorem.) The adjoint problem is

$$v'''' + \gamma^2 v = 0 , \tag{24}$$

with homogeneous boundary conditions

$$\begin{aligned} &v(-1) = 0 , \; v''(-1) = 0 \\ &v(1) = 0 , \; v''(1) = 0 , \end{aligned} \tag{25}$$

and the homogeneous continuity and jump conditions

$$
\begin{aligned}
&v(0^-) = v(0^+) \\
&v'(0^-) = v'(0^+) \\
&v''(0^-) = v''(0^+) \\
&v'''(0^+) - v'''(0^-) + kv(0) = 0\,,
\end{aligned}
\tag{26}
$$

with the compatibility condition

$$
pv(0) = 2\gamma^2\lambda\int_{-1}^{1} v(\bar{x})d\bar{x}\,. \tag{27}
$$

The adjoint boundary value problem is homogeneous, and has the trivial solution $v(\bar{x}) = 0$ for all $\bar{x}$. Note that the compatibility condition, eq. 27, is satisfied by the trivial solution for any p. Nontrivial solutions (eigenfunctions) for $v(\bar{x})$ exist if γ is an eigenvalue. If γ is an eigenvalue, then p, γ, and λ are related by the compatibility equation. For a given arch rise λ and eigenvalue γ, the load(s) p at which multiple solutions for $\bar{w}(\bar{x})$ are possible are determined by eq. 27. This adjoint problem has the same homogeneous solution as eq. 17, namely

$$
v = \begin{cases} B_1 \sin\gamma\bar{x} + B_2\cos\gamma\bar{x} + B_3\bar{x} + B_4 & \bar{x} \in (-1\,,0) \\ B_5 \sin\gamma\bar{x} + B_6\cos\gamma\bar{x} + B_7\bar{x} + B_8 & \bar{x} \in (0\,,1) \end{cases} \tag{28}
$$

Substituting eq. 28 into eqs. 25 and 26 results in a homogeneous problem for coefficients B_1 to B_8 with γ as the eigenvalue parameter. The nontrivial solutions to this problem are

$$
\begin{aligned}
&\text{A.} \quad \sin\gamma = 0 \quad (\gamma = n\pi\,,\ n = \text{positive integer}) \\
&\quad\ \text{A.1} \quad k \neq 2\gamma^2\,, \ \text{then} \\
&\qquad\qquad v_n(\bar{x}) = B_1 \sin\, n\pi\bar{x}\,, \quad \bar{x} \in (-1,1)
\end{aligned}
\tag{29}
$$

for which any value of p will satisty compatibility; and

A.2 $k = 2\gamma^2$, then

$$v_n(\bar{x}) = \begin{cases} B_\cdot \sin n\pi\bar{x} + B_3\left(-\frac{1}{n\pi}\sin n\pi\bar{x} + \bar{x} + 1\right), & \bar{x} \in (-1,0) \\ B_\cdot \sin n\pi\bar{x} + B_3\left(\frac{1}{n\pi}\sin n\pi\bar{x} - \bar{x} + 1\right), & \bar{x} \in (0,1) \end{cases} \quad (30)$$

for which

$$p = 4\lambda\left(1 - \cos(n\pi) + \frac{(n\pi)^2}{2}\right). \quad (31)$$

where

$$\gamma = \gamma_n = n\pi = \frac{\sqrt{k}}{2}, \quad \text{and} \quad B_\cdot = \frac{B_1 + B_5}{2} \quad (32)$$

B. $\sin\gamma \neq 0$

B.1 $k \neq 0$, then (33)

$k \sin\gamma + \gamma(2\gamma^2 - k)\cos\gamma = 0$ with

$$v(\bar{x}) = B_1 \begin{cases} \sin\gamma\bar{x} - \frac{2\gamma^2 - k}{k}\gamma\cos\gamma\bar{x} - \gamma\bar{x} - \gamma, & \bar{x} \in (-1,0) \\ -\sin\gamma\bar{x} - \frac{2\gamma^2 - k}{k}\gamma\cos\gamma\bar{x} + \gamma\bar{x} - \gamma, & \bar{x} \in (0,1) \end{cases}, \quad (34)$$

for which

$$p = \frac{-2\lambda k}{\gamma}\left(\frac{1}{\gamma}(\cos\gamma - 1) - \frac{2\gamma^2 - k}{k}\sin\gamma - \frac{\gamma}{2}\right), \quad (35)$$

B.2 $k = 0$, then

$\cos\gamma = 0$ with (36)

$v(\bar{x}) = B_2 \cos\gamma\bar{x}, \quad \bar{x} \in (-1,1)$

for which

$$p = 4\lambda\gamma\sin\gamma. \quad (37)$$

The solution for $\overline{w}$ when γ is an eigenvalue of the adjoint problem describes special but important system behavior. When γ is an eigenvalue the solution for $\overline{w}$ is still obtained from eqs. 22 and 23, however, C_{ij} is singular. In case A.1, C_{ij} is reduced to rank 7. The unknowns A_1, A_2, A_3, A_4, A_6, A_7, and A_8 can be solved for in eq. 22 in terms of A_5. The unknown A_5 is solved for in terms of p and λ in eq. 23. Solutions for A_5 exist only for values of λ greater than a threshold value and only for a range of p. For each p in this range there are two values of A_5, thus two solutions for $\overline{w}$. These solutions correspond to two asymmetric equilibrium paths bifurcating from the symmetric equilibrium path. Case A.2 is a special case of A.1. In this case C_{ij} is of rank 6 and A_2, A_3, A_4, A_6, A_7, and A_8 are solved for in terms of A_1 and A_5. The value of p for which solutions exist was determined from compatibility as given by eq. 31. Equation 23 is used to solve for A_5 in terms of A_1. Again solutions only exist for λ large enough. This case corresponds to bifurcation behavior in which the two bifurcation paths collapse to one solution in which the midspan deflection increases without a change in load. In cases B.1 and B.2, the matrix C_{ij} is again of rank 6 and in addition, $A_2 = A_6$. The remaining A's are solved for in terms of A_2, and A_2 is solved for in terms of λ via eq. 23. For given values of γ and k, related through eq. 33, there will be only one value of λ which yields a solution for A_2. This case describes the particular situation of a horizontal inflection point on the symmetric equilibrium path.

The equations presented in this section lead to the exact solution for the case of a shallow arch. As has been seen in past results [1,2], the response is complicated with a number of special but important cases. The governing equations will now be solved using a finite-difference formulation.

FINITE-DIFFERENCE REPRESENTATION OF THE INCREMENTAL FORM OF THE SHALLOW ARCH EQUATIONS

First-Order Form of Governing Equations

Using the governing equilibrium equations from the exact solution and the definitions

of shear force, and axial force and moment resultants, the following system of equations can be written:

$$\begin{aligned} &N' = 0 \\ &V' = 0 \\ &M' + N(z'_o + w') - V = 0 \\ &N = EA\left(u' + z'_o w' + \frac{1}{2}(w')^2\right) \\ &M = EI(-w'') . \end{aligned} \tag{38}$$

Introducing the additional definition

$$\beta = -w' \tag{39}$$

which is the rotation, produces a system of first order equations consisting of force and displacement variables. These equations are

$$\begin{aligned} &N' = 0 \\ &V' = 0 \\ &M' = -N(z'_o - \beta) + V \\ &u' = N/EA + z'_o\beta - \frac{1}{2}\beta^2 \\ &w' = -\beta \\ &\beta' = M/EI . \end{aligned} \tag{40}$$

Letting

$$\begin{aligned} y_1 &= N \qquad y_4 = u \\ y_2 &= V \qquad y_5 = w , \\ y_3 &= M \qquad y_6 = \beta \end{aligned} \tag{41}$$

the system of equations can be written as

$$\underset{\sim}{y}' = \underset{\sim}{\underset{\sim}{A}}\underset{\sim}{y} + f(\underset{\sim}{y}) \tag{42}$$

with boundary conditions

$$
\begin{aligned}
y_4(0) &= y_4(L) = 0 \\
y_5(0) &= y_5(L) = 0 \\
y_3(0) &= y_3(L) = 0
\end{aligned}
\tag{43}
$$

and continuity and jump conditions

$$
\begin{aligned}
y_1^- &= y_1^+ \\
y_2^- + P + K\bar{y}_5 &= y_2^+ \\
y_3^- &= y_3^+ \\
y_4^- &= y_4^+ \\
y_5^- &= y_5^+ \\
y_6^- &= y_6^+ ,
\end{aligned}
\tag{44}
$$

where the '+' and '-' denote values of the variables to the left and right of the center of the arch.

Considering each of the six equations in eq. 40 to be a function F_j, j = 1,6, of six variables, the governing system of first order equations can be written as

$$
F_j(N,\ V,\ M,\ u,\ w,\ \beta) = 0\,,\quad j = 1, 6\,. \tag{45}
$$

Finite-Difference Representation

Utilizing a finite-difference approximation for the 1st derivative, the governing equations become

$$
\begin{aligned}
F^k_{1i} &= \frac{N^k_{i+1} - N^k_i}{x_{i+1} - x_i} = 0 \\
F^k_{2i} &= \frac{V^k_{i+1} - V^k_i}{x_{i+1} - x_i} = 0 \\
F^k_{3i} &= \frac{M^k_{i+1} - M^k_i}{x_{i+1} - x_i} + \frac{1}{2}\left\{N^k_{i+1}\left(z'_{o\,i+1} - \beta^k_{i+1}\right) - V^k_{i+1}\right. \\
&\quad \left. + N^k_i\left(z'_{o\,i} - \beta^k_i\right) - V^k_i\right\} = 0 \\
F^k_{4i} &= \frac{u^k_{i+1} - u^k_i}{x_{i+1} - x_i} - \frac{1}{2}\left\{\frac{N^k_{i+1}}{EA} + z'_{o\,i+1}\beta^k_{i+1} - \frac{1}{2}\left(\beta^k_{i+1}\right)^2\right. \\
&\quad \left. + \frac{N^k_i}{EA} + z'_{o\,i}\beta^k_i - \frac{1}{2}\left(\beta^k_i\right)^2\right\} = 0 \\
F^k_{5i} &= \frac{w^k_{i+1} - w^k_i}{x_{i+1} - x_i} + \frac{1}{2}\left\{\beta^k_{i+1} + \beta^k_i\right\} = 0 \\
F^k_{6i} &= \frac{\beta^k_{i+1} - \beta^k_i}{x_{i+1} - x_i} - \frac{1}{2EI}\left\{M^k_{i+1} + M^k_i\right\} = 0
\end{aligned}
\tag{46}
$$

where i identifies values at the i-th grid point in the finite-difference grid and k signifies that the equations are being solved for the k-th load level. The boundary conditions are

$$
\begin{aligned}
M^k_1 &= 0 \qquad M^k_m = 0 \\
u^k_1 &= 0 \qquad u^k_m = 0 \\
w^k_1 &= 0 \qquad w^k_m = 0\,,
\end{aligned}
\tag{47}
$$

where the number of grid points is m, the 1-st grid point being the left boundary ($x = 0$) and the m-th grid point the right boundary ($x = L$). The continuity and jump conditions at the middle grid point are

$$
\begin{aligned}
N^k_{\frac{m}{2}+1} - N^k_{\frac{m}{2}} &= 0 \\
V^k_{\frac{m}{2}+1} - V^k_{\frac{m}{2}} - Kw_{\frac{m}{2}} &= P \\
M^k_{\frac{m}{2}+1} - M^k_{\frac{m}{2}} &= 0 \\
u^k_{\frac{m}{2}+1} - u^k_{\frac{m}{2}} &= 0 \\
w^k_{\frac{m}{2}+1} - w^k_{\frac{m}{2}} &= 0 \\
\beta^k_{\frac{m}{2}+1} - \beta^k_{\frac{m}{2}} &= 0 .
\end{aligned}
\tag{48}
$$

Incremental Form

Letting

$$
\begin{aligned}
N_i^{k+1} &= N_i^k + \Delta N_i^k \\
V_i^{k+1} &= V_i^k + \Delta V_i^k \\
M_i^{k+1} &= M_i^k + \Delta M_i^k \\
u_i^{k+1} &= u_i^k + \Delta u_i^k \\
w_i^{k+1} &= w_i^k + \Delta w_i^k \\
\beta_i^{k+1} &= \beta_i^k + \Delta \beta_i^k .
\end{aligned}
\tag{49}
$$

substituting into eq. 46, expanding, and neglecting higher-order terms, results in the following set of equations for the increments in the six variables:

$$F_{1i}^{k+1} = \frac{N_{i+1}^k - N_i^k}{x_{i+1} - x_i} + \frac{\Delta N_{i+1}^k - \Delta N_i^k}{x_{i+1} - x_i} = 0$$

$$F_{2i}^{k+1} = \frac{V_{i+1}^k - V_i^k}{x_{i+1} - x_i} + \frac{\Delta V_{i+1}^k - \Delta V_i^k}{x_{i+1} - x_i} = 0$$

$$F_{3i}^{k+1} = \frac{M_{i+1}^k - M_i^k}{x_{i+1} - x_i} + \frac{1}{2}\left(z'_{oi+1}N_{i+1}^k - \beta_{i+1}^k N_{i+1} - V_{i+1}^k + z'_{oi}N_i^k - \beta_i^k N_i^k - V_i^k\right)$$
$$+ \frac{\Delta M_{i+1}^k - \Delta M_i^k}{x_{i+1} - x_i} + \frac{1}{2}\left(z'_{oi+1}\Delta N_{i+1}^k - \beta_{i+1}^k \Delta N_{i+1}^k - \Delta\beta_{i+1}^k N_{i+1}^k - \Delta V_{i+1}^k\right.$$
$$\left. + z'_{oi}\Delta N_i^k - \beta_i^k \Delta N_i^k - \Delta\beta_i^k N_i^k - \Delta V_i^k\right) = 0$$

$$F_{4i}^{k+1} = \frac{u_{i+1}^k - u_i^k}{x_{i+1} - x_i} - \frac{1}{2}\left\{\frac{1}{EA}N_{i+1}^k + z'_{oi+1}\beta_{i+1}^k - \frac{1}{2}\left(\beta_{i+1}^k\right)^2\right. \tag{50}$$
$$\left. + \frac{1}{EA}N_i^k + z'_{oi}\beta_i^k - \frac{1}{2}\left(\beta_i^k\right)^2\right\}$$
$$+ \frac{\Delta u_{i+1}^k - \Delta u_i^k}{x_{i+1} - x_i} - \frac{1}{2}\left\{\frac{1}{EA}\Delta N_{i+1}^k + z'_{oi+1}\Delta\beta_{i+1}^k - \beta_{i+1}^k \Delta\beta_{i+1}^k\right.$$
$$\left. + \frac{1}{EA}\Delta N_i^k + z'_{oi}\Delta\beta_i^k - \beta_i^k \Delta\beta_i^k\right\} = 0$$

$$F_{5i}^{k+1} = \frac{w_{i+1}^k - w_i^k}{x_{i+1} - x_i} + \frac{1}{2}\left(\beta_{i+1}^k + \beta_i^k\right) + \frac{\Delta w_{i+1}^k - \Delta w_i^k}{x_{i+1} - x_i} + \frac{1}{2}\left(\Delta\beta_{i+1}^k + \Delta\beta_i^k\right) = 0$$

$$F_{6i} = \frac{\beta_{i+1}^k - \beta_i^k}{x_{i+1} - x_i} - \frac{1}{2EI}\left(M_{i+1}^k + M_i^k\right) + \frac{\Delta\beta_{i+1}^k - \Delta\beta_i^k}{x_{i+1} - x_i} - \frac{1}{2EI}\left(\Delta M_{i+1}^k - \Delta M_i^k\right) = 0,$$

with boundary conditions

$$\begin{aligned} M_1^k + \Delta M_1^k &= 0 \\ u_1^k + \Delta u_1^k &= 0 \\ w_1^k + \Delta w_1^k &= 0 \\ M_m^k + \Delta M_m^k &= 0 \\ u_m^k + \Delta u_m^k &= 0 \\ w_m^k + \Delta w_m^k &= 0 \end{aligned} \tag{51}$$

and continuity and jump conditions

$$
\begin{aligned}
&N^k_{\frac{m}{2}+1} - N^k_{\frac{m}{2}} + \Delta N^k_{\frac{m}{2}+1} + \Delta N^k_{\frac{m}{2}} = 0 \\
&V^k_{\frac{m}{2}+1} - V^k_{\frac{m}{2}} - Kw^k_{\frac{m}{2}} + \Delta V^k_{\frac{m}{2}+1} - \Delta V^k_{\frac{m}{2}} - K\Delta w^k_{\frac{m}{2}} = P \\
&M^k_{\frac{m}{2}+1} - M^k_{\frac{m}{2}} + \Delta M^k_{\frac{m}{2}+1} - \Delta M^k_{\frac{m}{2}} = 0 \\
&u^k_{\frac{m}{2}+1} - u^k_{\frac{m}{2}} + \Delta u^k_{\frac{m}{2}+1} - \Delta u^k_{\frac{m}{2}} = 0 \\
&w^k_{\frac{m}{2}+1} - w^k_{\frac{m}{2}} + \Delta w^k_{\frac{m}{2}+1} - \Delta w^k_{\frac{m}{2}} = 0 \\
&\beta^k_{\frac{m}{2}+1} - \beta^k_{\frac{m}{2}} + \Delta\beta^k_{\frac{m}{2}+1} - \Delta\beta^k_{\frac{m}{2}} = 0 .
\end{aligned}
\tag{52}
$$

Specifically, we have equations which can be written as

$$
[A]\{\Delta y\} = \{R\} \tag{53}
$$

If the coefficient matrix [A], is not singular, the equations can be solved for the increments in the force and displacement variables. A solution is realized when the residual vector {R} is zero. This condition is achieved by iteration at a given load level.

At limit and bifurcation points the coefficient matrix [A] becomes singular, and the eigenvalues of [A] are used to evaluate the singular nature. As can be seen from eq. 52, the [A] matrix is not a classic stiffness matrix (i.e., it is not symmetric) since it involves both forces and displacements. Its eigenvalues are not all real and vary greatly in magnitude which makes it difficult to analyze the singular nature of the matrix and use the eigenvalues as an indication of singular behavior. Equation 53 can be partitioned as

$$
\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{Bmatrix} \Delta F \\ \Delta u \end{Bmatrix} = \begin{Bmatrix} R_1 \\ R_2 \end{Bmatrix} \tag{54}
$$

where ΔF are the force variables ΔN, ΔV, ΔM, and Δu are the displacement variables Δw, Δu, $\Delta \beta$. It was hoped that by eliminating the force variables from eq. 54 a classic stiffness matrix would result. Doing so led to an equation of the form

$$[A^*]\{\Delta u\} = \{R^*\} \tag{55}$$

where

$$[A^*] = [A_{11} - A_{21}\ A_{11}^{-1}\ A_{12}] \tag{56}$$

and

$$\{R^*\} = \{R_2 - A_{21}\ A_{11}^{-1}\ R_1 .\} \tag{57}$$

Unfortunately the $[A^*]$ matrix was not symmetric either and could not be used to help analyze or indicate the singular nature of the problem at bifurcation points and limit points.

Dynamic Stability Analysis

As a final approach, it was hoped a dynamic stability analysis would aid in determining the bifurcation and limit points. For the dynamic problem, the governing equations are

$$\begin{aligned} N_{,x} &= 0 \\ V_{,x} &= m w_{,tt} \\ M_{,x} &= -N(z'_o - \beta) + V \\ u_{,x} &= N/EA + z'_o\beta - \frac{1}{2}\beta^2 \\ w_{,x} &= -\beta \\ \beta_{,x} &= M/EI\ , \end{aligned} \tag{58}$$

where variables N, V, M, u, w, β, - i.e., y_i, i = 1,6, - are now functions of space and time. It is assumed that

$$\underset{\sim}{y}(x,t) = \underset{\sim}{y}_e(x) + \Delta\underset{\sim}{y}(x,t)\,, \tag{59}$$

the two parts on the right hand side representing an equilibrium part and a dynamic part to the solution. Further assuming that

$$\Delta\underset{\sim}{y}(x,t) = \Delta\underset{\sim}{y}_s(x)e^{\lambda t}$$

and using the finite-difference formulation on eq. 58, the system of equations for the Δy_s results. This system is of the form

$$[A]\{\Delta\underset{\sim}{y}_s\} - \lambda^2[B]\{\Delta\underset{\sim}{y}_s\} = 0 \tag{60}$$

from which the eigenvalues λ can be determined. It was found that the eigenvalues of the dynamic stiffness matrix were also complex. Thus the dynamic stiffness analysis did not offer any advantage over the examination of the eigenvalues of the original A matrix in eq. 53.

COMPARISON OF CLOSED-FORM AND FINITE-DIFFERENCE SOLUTIONS FOR THE SHALLOW ARCH

Several important cases of shallow arch response were studied with the finite-difference formulation, and the results were compared with the closed-form solution. The load-midspan deflection relation of a spring-reinforced arch with $\lambda = 1.5$ and $k = 2$ is shown in fig. 3. These values of the parameters result in limit point behavior. Both the closed-form solution, the dashed line, and the finite-difference solution evaluated at distinct load levels, the asterisks, are shown. The numerical values were generated by increasing the load from zero and proceeding to the load level represented by the asterisk just to the lower left of the limit point. At this point the eigenvalue strategy was used to move to a point represented by the asterisk just to the lower right of the limit point. The load level was then decreased and the solution continued. Moving the numerical solution past the second limit point, point L′, was accomplished

in a similar manner. Without the eigenvalue strategy, it would not be possible to move the numerical solution past either limit point. The numerical solution could be made to have as many discrete points as desired, the number shown in fig. 3 being selected simply for purposes of illustration. The load-thrust relation for this same case is shown in fig. 4.

Figure 5 illustrates the response of an arch with the parameters chosen so it exhibits bifurcation behavior as the load increases from zero. For this case, $\lambda = 2.5$ and $k = 10$. At point C, the load-deflection relation can either continue with the load increasing to the limit point, or the relation can branch to a secondary path, the load decreasing with increasing deflections. Whereas the solution path from zero load to the limit point represents arch response that is symmetric with respect to the midspan, the bifurcated solution path represents arch response that is not symmetric with respect to midspan. To be forewarned that the solution was about to bifurcate, the eigenvalues of the coefficient matrix were computed as the load increased from zero. For this problem there were real as well as complex eigenvalues. Fortunately, one of the real eigenvalues tending to zero provided an indication that a bifurcation point was being approached. There is no guarantee with the mixed formulation that any of the eigenvalues have to be real and provide an indication of the impending bifurcation. This, as mentioned at the onset, represents an important disadvantage to the mixed approach.

The move onto the secondary path was accomplished with the eigenvector associated with the real eigenvalue that did approach zero. If the eigenvector was not used, the solution would continue on the primary path and the eigenvector continuation approach could be used to move past the limit point, as in fig. 3. If the solution was on the secondary path, the move back onto the primary path at D was no particular problem. The load-thrust relation for this case is illustrated in fig. 6. The interesting feature to note is that if the response is on the secondary path, CD, the thrust remains

constant even though the applied load decreases as the arch deforms. On the primary path both the thrust and the load change values as the arch deforms. It is also important to note for the case shown in figs. 5 and 6 that to move onto the secondary path, the load must decrease. If the load increases, the response remains on the primary path until the limit point. At this point, under load control the deflections then jumps to the remaining portion of the primary path, as in fig. 2a. If under displacement control at the limit point, the load must decrease for the displacements to increase. As can be seen, the finite-difference with the continuation method and the closed-form solution agree perfectly.

To have the ability to increase the load, yet not experience the snap-through at the limit point, the behavior shown in fig. 7 is desirable. This figure illustrates the correlation between finite-difference solution and the closed-form solution, but it also illustrates a useful response. If the spring stiffness is increased so $k = 25$, the arch midspan deflection behaves as shown in the figure. As the load is increased from zero, a bifurcation is encountered at C. The response can move to the bifurcated path and the load can continue to increase, with moderate increases in deflection. Thus, with the proper choice of spring stiffness, neither the decrease in load required in fig. 5 and 6, nor the sudden jump in displacement due to limit point behavior have to be tolerated. This is a significant finding, one that has important physical implications. As noted in fig. 8, on the bifurcated path the thrust is not influenced by the load level. This also has important ramifications.

Despite the success of the finite-difference approach coupled with the continuation method, as indicated by the excellent agreement of the last several figures, the method was not based on principles that would guarantee success with all problems encountered. Hence the entire problem was reformulated with a displacement-based finite-element approach. With a displacement-based approach, the coefficient matrix that results would be symmetric and thus its eigenvalues real. In addition, positive

definiteness of the matrix, or the lack thereof, can be used to study stability of the response as the solution bifurcated. As an alternative, the classic dynamic stability approach could be used. The following section outlines the finite-element formulation.

FINITE-ELEMENT FORMULATION FOR A GENERAL ARCH

Basic Definitions

The finite-element approach was formulated using the Principle of Virtual Work. The Principle of Virtual Work requires that the internal work of a system equal the external work, i.e.,

$$\delta W_{int} = \delta W_{ext} \tag{61}$$

The internal and external virtual work expressions for a general arch with a center-span load and linear spring are developed in detail in Appendix A. The basic assumptions used to develop the virtual work expressions are that the strain of the reference arc is small compared to unity, the Kirchhoff-Love hypotheses govern the strain of parallel arcs, the rotation and rotation gradients as given by inextensional theory are sufficiently accurate for a small strain (extensional) theory, the material is linear elastic and that the normal stress components in Hooke's law can be neglected with respect to the hoop normal stress, and that the reference arc passes through the centroid of each cross section. The virtual work expressions are (see also the summary of the deep arch in Appendix A).

$$\delta W_{int} = \int_0^S \delta \underline{\varepsilon}^T \underline{\sigma} \, ds_o \, .$$

and (62)

$$\delta W_{ext} = P\delta w_m + K(\ell - \ell_o)\left[\frac{(\ell_o - w_m)}{\ell} \delta w_m - \frac{u_m}{\ell} \delta u_m \right]$$

where

$$\ell = \sqrt{(\ell_o - w_m)^2 + u_m^2} \ . \tag{63}$$

The quantity ℓ_o is the original length of the initially vertical spring and ℓ is the length after deformation. In the expression for internal work, the generalized strain vector is

$$\underset{\sim}{\varepsilon} = \begin{Bmatrix} \varepsilon_o \\ \beta' \end{Bmatrix} \tag{64}$$

in which the prime means derivative with respect to arc length coordinate s_o, and the stress vector is

$$\underset{\sim}{\sigma} = \begin{Bmatrix} N \\ M \end{Bmatrix} = \underset{\sim}{C}\underset{\sim}{\varepsilon} \ , \tag{65}$$

where the elasticity matrix is:

$$C = \begin{bmatrix} E\bar{A} & -E\bar{I}\kappa_o \\ -E\bar{I}\kappa_o & E\bar{I} \end{bmatrix} . \tag{66}$$

In the elasticity matrix the following definitions are used:

$$\bar{A} = \int\int_A \frac{1}{1-\zeta\kappa_o}\, dA \qquad \text{and} \qquad \bar{I} = \int\int_A \frac{\zeta^2}{1-\zeta\kappa_o}\, dA \tag{67}$$

where the κ_o is the curvature of the initial configuration and ζ is the thickness coordinate measured from the centroid. Expanding, the strains can be written in terms of the displacement gradients as

$$\varepsilon_o = \Gamma_T + \frac{1}{2}\Gamma_T^2 + \frac{1}{2}\Gamma_N^2 \quad \text{and} \quad \beta' = \frac{\Gamma'_N}{\sqrt{1-\Gamma_N^2}} \,. \tag{68}$$

The displacement gradients are defined by

$$\Gamma_T = u' - \kappa_o w \quad \text{and} \quad \Gamma_N = w' + \kappa_o u \,. \tag{69}$$

(Note that the "o" subscripts on the tangential and normal displacements of the reference arc used in Appendix A have been dropped here for convenience.) The variation in the strains are

$$\begin{aligned} \delta\varepsilon_o &= (1 + \Gamma_T)\delta\Gamma_T + \Gamma_N\delta\Gamma_N \\ \delta\beta' &= \frac{\delta\Gamma'_N}{\sqrt{1-\Gamma_N^2}} + \frac{\Gamma_N\Gamma'_N}{(1-\Gamma_N^2)^{3/2}}\delta\Gamma_N \,. \end{aligned} \tag{70}$$

Incremental Form

Using an incremental formulation to solve for the unknown displacements, the substitution

$$u \to u + \Delta u \quad \text{and} \quad w \to w + \Delta w \tag{71}$$

is made, and all dependent variables are linearized in the increments. Thus

$$\Gamma_T = \Gamma_T + \Delta\Gamma_T \quad \text{and} \quad \Gamma_N = \Gamma_N + \Delta\Gamma_N \,, \tag{72}$$

where

$$\Delta\Gamma_T = \Delta u' - \kappa_o \Delta w \quad \text{and} \quad \Delta\Gamma_N = \Delta w' + \kappa_o \Delta u \,. \tag{73}$$

The incremental strains are

$$\varepsilon_o = \varepsilon_o + \Delta\varepsilon_o \quad \text{and} \quad \beta' = \beta' + \Delta\beta' \,, \tag{74}$$

where

$$\Delta\varepsilon_o = (1 + \Gamma_T)\Delta\Gamma_T + \Gamma_N \Delta\Gamma_N$$
$$\Delta\beta' = \frac{\Delta\Gamma'_N}{\sqrt{1-\Gamma_N^2}} + \frac{\Gamma_N\Gamma'_N}{(1-\Gamma_N^2)^{3/2}}\Delta\Gamma_N . \tag{75}$$

Incrementing the actual displacements in the strain variations results in

$$\delta\varepsilon_o = (1+\Gamma_T)\delta\Gamma_T + \Gamma_N\delta\Gamma_N + \Delta\Gamma_T\delta\Gamma_T + \Delta\Gamma_N\delta\Gamma_N$$
$$\delta\beta' = \frac{\delta\Gamma'_N}{\sqrt{1-\Gamma_N^2}} + \frac{\Gamma_N\Gamma'_N\delta\Gamma_N}{(1-\Gamma_N^2)^{3/2}} + \frac{\Gamma_N\Delta\Gamma_N}{(1-\Gamma_N^2)^{3/2}}\delta\Gamma'_N$$
$$+ \left\{\frac{\Gamma_N\Delta\Gamma'_N}{(1-\Gamma_N^2)^{3/2}} + \frac{\Gamma'_N(1+2\Gamma_N^2)}{(1-\Gamma_N^2)^{5/2}}\Delta\Gamma_N\right\}\delta\Gamma_N . \tag{76}$$

Introduction of Finite-Elements

At this point the displacement interpolations for a seven degree-of-freedom element with three degrees of freedom in the tangential displacement - one at each end of the element and one at the center - and four degrees of freedom associated with the normal displacement w and w′ are introduced. The result is that

$$u = \underline{H}_1\hat{\underline{u}}, \quad \Delta u = \underline{H}_1\Delta\hat{\underline{u}}, \quad \delta u = \underline{H}_1\delta\hat{\underline{u}}$$
$$w = \underline{H}_2\hat{\underline{u}}, \quad \Delta w = \underline{H}_2\Delta\hat{\underline{u}}, \quad \delta w = \underline{H}_2\delta\hat{\underline{u}} . \tag{77}$$

In the above the seven nodal displacements are

$$
\hat{u} = \begin{Bmatrix} u_1 \\ w_1 \\ w'_1 \\ u_2 \\ w_2 \\ w'_2 \\ u_3 \end{Bmatrix} \tag{78}
$$

and the shape functions are

$$
\begin{aligned}
&\underline{H}_1 = [H_{11}(s_o),\ 0,\ 0,\ H_{14}(s_o),\ 0,\ 0,\ H_{17}(s_o)]\ , \\
\text{and} \qquad & \\
&\underline{H}_2 = [0,\ H_{22}(s_o),\ H_{23}(s_o),\ 0,\ H_{25}(s_o),\ H_{26}(s_o),\ 0]
\end{aligned} \tag{79}
$$

with

$$
\begin{aligned}
H_{11}(s_o) &= \frac{\left(\dfrac{2(s_o - s_{o1})}{h_{el}} - 1\right)^2}{2} - \frac{\left(\dfrac{2(s_o - s_{o1})}{h_{el}} - 1\right)}{2} \\
H_{14}(s_o) &= \frac{\left(\dfrac{2(s_o - s_{o1})}{h_{el}} - 1\right)^2}{2} + \frac{\left(\dfrac{2(s_o - s_{o1})}{h_{el}} - 1\right)}{2} \\
H_{17}(s_o) &= 1 - \left(\frac{2(s_o - s_{o1})}{h_{el}} - 1\right)^2, \qquad s_{o1} \leq s_o \leq s_{o2}
\end{aligned} \tag{80}
$$

and

$$
\begin{aligned}
H_{22}(s_o) &= 1 - 3\left(\frac{s_o - s_{o1}}{h_{el}}\right)^2 + 2\left(\frac{s_o - s_{o1}}{h_{el}}\right)^3 \\
H_{23}(s_o) &= -(s_o - s_{o1})\left(1 - \frac{s_o - s_{o1}}{h_{el}}\right)^2 \\
H_{25}(s_o) &= 3\left(\frac{s_o - s_{o1}}{h_{el}}\right)^2 - 2\left(\frac{s_o - s_{o1}}{h_{el}}\right)^3 \\
H_{26}(s_o) &= -(s_o - s_{o1})\left(\left(\frac{s_o - s_{o1}}{h_{el}}\right)^2 - \left(\frac{s_o - s_{o1}}{h_{el}}\right)\right), \qquad s_{o1} \le s_o \le s_{o2}
\end{aligned}
\tag{81}
$$

where s_{o1} is the arc-length location of the left end of the element, s_{o2} is the arc length location of the right end, and h_{el} is the arc length of the element. See Fig. 9 for a sketch of the element. Substituting the shape functions into the strain and strain variations leads to

$$
\delta\varepsilon_o = \underline{B}_1(\hat{\underline{u}})\delta\hat{\underline{u}}\,, \tag{82}
$$

where $\underline{B}_1(\hat{\underline{u}})$ is a 1 x 7 matrix given by

$$
\begin{aligned}
\underline{B}_1 = \underline{B}_1(\hat{u}) = \underline{H}'_1 - \kappa_o\underline{H}_2 &+ \hat{\underline{u}}^T(\underline{H}'_1 - \kappa_o\underline{H}_2)^T(\underline{H}'_1 - \kappa_o\underline{H}_2) \\
&+ \hat{\underline{u}}^T(\underline{H}'_2 + \kappa_o\underline{H}_1)^T(\underline{H}'_2 + \kappa_o H_1)\,,
\end{aligned}
\tag{83}
$$

and

$$
\delta\beta' = \underline{B}_2(\hat{\underline{u}})\delta\hat{\underline{u}}\,, \tag{84}
$$

where $\underline{B}_2(\hat{\underline{u}})$ is a 1 x 7 matrix given by

$$
\begin{aligned}
\underline{B}_2 = \underline{B}_2(\hat{\underline{u}}) = &\frac{1}{\sqrt{1 - \Gamma_N^2}}(\underline{H}''_2 + \kappa_o\underline{H}'_1) \\
&+ \frac{\Gamma_N\Gamma'_N}{(1 - \Gamma_N^2)^{3/2}}(\underline{H}'_2 + \kappa_o\underline{H}_1)\,.
\end{aligned}
\tag{85}
$$

In the above

$$\Gamma_N^2 = \hat{\underset{\sim}{u}}^T(\underline{H}'_2 + \kappa_o\underline{H}_1)^T(\underline{H}'_2 + \kappa_o\underline{H}_1)\hat{\underset{\sim}{u}} \tag{86}$$

$$\Gamma_N\Gamma'_N = \underset{\sim}{u}^T(\underline{H}'_2 + \kappa_o\underline{H}_1)^T(\underline{H}''_2 + \kappa_o\underline{H}'_1)\hat{\underset{\sim}{u}}\,. \tag{87}$$

Also the incremental strain and incremental displacement relations are

$$\Delta\varepsilon_o = \underline{B}_1\Delta\hat{\underset{\sim}{u}} \quad \text{and} \quad \Delta\beta' = \underline{B}_2\Delta\hat{\underset{\sim}{u}}\,. \tag{88}$$

Incrementing the actual displacements in the strain variations, these incremental strain variations can be written as

$$\delta\varepsilon_o = \underline{B}_1(\hat{\underset{\sim}{u}})\delta\hat{\underset{\sim}{u}} + \Delta\underset{\sim}{u}^T\underline{D}\delta\hat{\underset{\sim}{u}} \quad \text{and} \quad \delta\beta' = B_2(\hat{\underset{\sim}{u}})\delta\hat{\underset{\sim}{u}} + \Delta\underset{\sim}{u}^T\underline{E}(\hat{\underset{\sim}{u}})\delta\hat{\underset{\sim}{u}}\,, \tag{89}$$

where

$$\underline{D} = (\underline{H}'_1 - \kappa_o\underline{H}_2)^T(\underline{H}'_1 - \kappa_o\underline{H}_2) + (\underline{H}'_2 + \kappa_o\underline{H}_1)^T(\underline{H}'_2 + \kappa_o\underline{H}_1) \tag{90}$$

and

$$\begin{aligned}\underline{E}(\hat{\underset{\sim}{u}}) = {} & \frac{\Gamma_N}{(1-\Gamma_N^2)^{3/2}}\left[(\underline{H}'_2 + \kappa_o\underline{H}_1)^T(\underline{H}''_2 + \kappa_o\underline{H}'_1) + (\underline{H}''_2 + \kappa_o\underline{H}'_1)^T(\underline{H}'_2 + \kappa_o\underline{H}_1)\right] \\ & + \frac{\Gamma'_N(1+2\Gamma_N^2)}{(1-\Gamma_N^2)^{5/2}}\left[(\underline{H}'_2 + \kappa_o\underline{H}_1)^T(\underline{H}'_2 + \kappa_o\underline{H}_1)\right].\end{aligned} \tag{91}$$

Note that the 7×7 matricies $\underline{D}$ and $\underline{E}$ are symmetric. Using the expressions for the incremental strain variation, the strain variation, and Hooke's law, the internal virtual work for an element can be written

$$\delta W_{int} = \delta\hat{\underset{\sim}{u}}^T\underline{R} + \delta\hat{\underset{\sim}{u}}^T[\underline{K} + \underline{K}_G]\Delta\hat{\underset{\sim}{u}}\,. \tag{92}$$

In this expression

$$\underline{R} = \int_{h_{el}} \begin{bmatrix} \underline{B}_1^T & \underline{B}_2^T \end{bmatrix} \begin{bmatrix} E\bar{A} & -E\bar{I}\kappa_o \\ -E\bar{I}\kappa_o & E\bar{I} \end{bmatrix} \begin{bmatrix} \underline{B}_1 \\ \underline{B}_2 \end{bmatrix} \hat{\underline{u}}\, ds_o$$
$$= \int_{h_{el}} \underline{B}^T \underline{C} \underline{B} \hat{\underline{u}}\, ds_o\,, \tag{93}$$

$$\underline{K} = \int_{h_{el}} \underline{B}^T \underline{C} \underline{B}\, ds_o\,, \tag{94}$$

and

$$\underline{K}_G = \int_{h_{el}} \left(\underline{C} \underline{B}_1 \hat{\underline{u}} \underline{D} + \underline{C} \underline{B}_2 \hat{\underline{u}} \underline{E} \right) ds_o\,. \tag{95}$$

Here $\underline{K} + \underline{K}_G$ is referred to as the tangential stiffness matrix. It is evaluated numerically, specifically, by using Simpson's integration rule. An issue is the number of intervals to be used to evaluate the integral. This is addressed in the next section. The actual displacements in the external work are also incremented to give

$$\begin{aligned} \delta W_{ext} = {} & P \delta w_m \\ & + \{Q_N - K_{NN}\Delta w_m - K_{NT}\Delta u_m\}\delta w_m \\ & + \{Q_T - K_{TN}\Delta w_m - K_{TT}\Delta u_m\}\delta u_m \end{aligned} \tag{96}$$

in which the incremental quantities from the spring are

$$\begin{aligned} Q_N &= F_S(\ell_o - w_m)/\ell \\ Q_T &= -F_S(u_m/\ell) \\ K_{NN} &= K\left(\frac{\ell_o - w_m}{\ell}\right)^2 + \frac{F_S}{\ell}\left[1 - \left(\frac{\ell_o - w_m}{\ell}\right)^2\right] \\ K_{NT} = K_{TN} &= -K\left(\frac{u_m}{\ell}\right)\left(\frac{\ell_o - w_m}{\ell}\right) + F_S\left(\frac{u_m}{\ell}\right)\left(\frac{\ell_o - w_m}{\ell}\right) \\ K_{TT} &= K\left(\frac{u_m}{\ell}\right)^2 + \frac{F_S}{\ell}\left[1 - \left(\frac{u_m}{\ell}\right)^2\right] \end{aligned} \tag{97}$$

The incremental expression for the internal virtual work can then be set equal to the incremental expression for the external virtual work to solve for the displacements in an incremental fashion.

This completes the outline of the finite-element formulation. It is totally displacement-based and hence the eigenvalues of the stiffness matrix are expected to be useful for the analysis of instability.

COMPARISON OF CLOSED-FORM AND FINITE-ELEMENT APPROACHES FOR THE SHALLOW ARCH

Initial Behavior

Before implementing the eigenvalue continuation scheme within the finite element method the inital response of several shallow arches, beginning from zero load and displacement, was compared using the finite element method and the closed-form solution. In fig. 10 the comparison for one of the shallow arches is illustrated. Four elements are used and four intervals in the application of the Simpson's rule are used to approximate the integrals in the tangent stiffness matrix, eqs. 94 and 95. This case has no spring, $k = 0$, and a value of the arch rise parameter $\lambda = 0.7656$ which produces load-midspan deflection behavior that is monotonically increasing everywhere along the path. There is no bifurcation or limit point for this case. Clearly, the finite-element analysis and the closed-form solution are in complete agreement. In fig. 11 comparison is made for the value of $\lambda = 1.5$, k again being zero (no spring). This value of λ results in limit point behavior. As can be seen, at the limit point the finite-element analysis responds to an increasing load level by jumping to the other portion of the solution path. In fig. 12 a case of bifurcating behavior is shown, the value of λ being 4, k still being zero. In the finite-element analysis, as the load is increased from zero and the bifurcation encountered, the solution responds by remaining on the primary load path. For this case the load was not increased beyond the limit load.

The finite-element formulation and closed-form results agree very well for the initial response (response before limit point) of the shallow arch. The influence of more elements or more integration intervals was not explored because of the good agreement with what was felt to be a rather crude model.

Behavior with Singular Points

To be able to continue solutions through limit points and to branch on to bifurcated paths, an eigenvalue continuation method, similar to that used with the finite-difference scheme described earlier, was implemented in the finite-element analysis. As a bifurcation point or limit point is approached, one of the eigenvalues of the tangent stiffness matrix will approach and pass through zero. In the displacement based finite-element formulation the eigenvalues of the tangent stiffness matrix are all real and thus the existence of a zero eigenvalue clearly indicated the singular nature of the tangent stiffness matrix at bifurcation and limit points. At a bifurcation point the eigenvector associated with the zero eigenvalue is asymmetric. If the solution at a load just prior to the bifurcation point is modified by adding to it this eigenvector (normalized) and multiplied by a scale factor, a solution on the bifurcated path can be found. Once on the bifurcated path, the usual solution technique can be used to continue along the path. At a limit point the eigenvector associated with the zero eigenvalue is symmetric. If the scaled eigenvector is added to the solution at a point just before the limit point, a point on the path just after the limit point can be determined. As with the bifurcated path, once a solution past the limit point is found, the arch response can be followed using the standard solution technique.

Figures 13, 14, and 15 show the agreement between the finite-element solution and the closed-form solution when the eigenvalue continuation scheme is used in the vicinity of bifurcation and limit points. Figure 13 illustrates the case with nondimensional spring stiffness $k = 2$ and $\lambda = 1.5$. With these parameters the arch will exhibit limit point behavior with no bifurcations. The load-midspan deflection relation, be-

ginning at zero load, is determined by using the solution at the previous load as the initial guess to the solution at the next load step. This process is repeated until the limit point is encountered. Using the solution at the point just to the lower left of the limit point, the eigenvalue continuation scheme is used to determine the solution at the point just to the right of the limit point. From this point the load level is decreased stepwise and the remainder of the relationship is generated, each load step using the solution at the previous load step as an initial guess. The notation 'batch' and 'interactive' in fig. 13 refers to the fact that two computer codes were written for the finite-element formulation with the continuation. The batch program automates somewhat the steps necessary to continue the solution past a limit point. The interactive program requires user intervention and allows the user to vary the parameters associated with the continuation scheme based on the nature of the solution, the eigenvalues, and the eigenvectors.

Figure 14 illustrates the agreement between the finite-element and closed-form solutions for an arch with $\lambda = 2.5$ and $k = 10$, an arch which exhibits both limit point and bifurcation behavior. At the bifurcation point the solution can continue along the symmetric path using the standard solution technique, or the solution on the bifurcated path can be determined using the eigenvalue continuation scheme. Note that the eigenvector associated with the eigenvalue that approaches zero at the bifurcation point is asymmetric. At the limit point the eigenvalue continuation scheme can be used to move past the limit point. Again, once a point on the bifurcated path is found, or a solution past the limit point is found, the standard solution technique can be used to follow the arch response.

Figure 15 shows the agreement between the finite-element and closed-form solutions for a shallow arch that exhibits bifurcation behavior but no limit point. The parameters for this arch are $\lambda = 2.5$ and $k = 70$. For this arch the eigenvalue continuation

scheme is necessary only to get onto the bifurcated path, the symmetric path can be followed using the standard solution technique.

In summary, for the shallow arch, the finite-element formulation and the closed-form results agree very well, both on primary paths and on adjacent equilibrium paths, which are reached with the finite-element method by using the eigenvalue continuation scheme. All the arches which used the eigenvalue continuation method were modeled with 16 elements and 8 integration intervals. The influence of the number of elements or the number of integration intervals was not investigated due to the good agreement with the closed form solution.

COMPARISONS FOR NONSHALLOW ARCH USING FINITE-ELEMENT APPROACH

Huddleston [3] published a series of papers dealing with the response of deep arches. Huddleston obtained numerical results from his first-order formulation by using the so-called shooting method. With this method, a boundary value problem is converted to an initial value problem. Using conditions at one boundary, a predictor-corrector method is used to integrate the governing differential equations and compute the conditions at the other boundary. If the conditions at the second boundary do not match what the boundary conditions there should actually be, the initial boundary conditions are adjusted, and the process repeated. The process is repeated until the boundary conditions on both ends of the arch match the desired conditions. To compare with Huddleston, nomenclature peculiar to that formulation must be introduced. Specifically,

$$I' = \iint \frac{y^2}{1 - \kappa_o y}\, dA, \tag{98}$$

which is a modified second moment of the cross-sectional area, where

$$\kappa_o = \text{initial curvature of arch.} \tag{99}$$

Quantity I′ in eq. (98) is identical to the quantity $\bar{I}$ in eq. (67). Also

$$\Delta = \text{midspan displacement}$$

$$\text{DELV} = \frac{\Delta}{L} = \text{nondimensional displacement}$$

$$Q = \frac{PL^2}{EI'} = \text{nondimensional load} \tag{100}$$

$$\text{CRUX} = \frac{I'}{AL_2} = \text{compressibility parameter}$$

$$\text{REL} = H/L = \text{height ratio}.$$

Since Huddleston reported the numerical values of the nondimensional parameters, there are wide choices of values for the physical dimensions of the arch that can be used and still duplicate the values of those nondimensional parameters. A comparison with Huddleston for the case of CRUX = 0 and REL = 0.25 is shown in fig. 16. (Note, to have CRUX = 0 the arch has to be infinitely thin, so that the radius of gyration, I/A , is zero. Alternatively, the arch has to be infinitely long. These are both extremes and cannot be duplicated exactly by the finite-element formulation.) In fig. 16 two finite-element descretizations, 16 elements and 32 elements, two levels of Simpson's rule integration accuracy, four intervals and eight intervals, and two absolute arch thickness are included in the figure. The value of L was chosen to be 16 in. The material considered was aluminum, with Young's modulus of 10 Msi and Poisson's ratio of 0.3. The results from Huddleston were obtained by interpolating from the figures in ref. 3. Initially, to compare with Huddleston, 16 elements and four integration intervals were used, and the arch thickness, h, was chosen to be 0.1 in. This led to a value for CRUX of 0.32×10^{5}. The calculations with these parameters are represented by the open squares. The finite-element calculations were stiff relative to Huddleson's results, the asterisks. To overcome this, the finite-element results were computed for the case of a thinner arch, h = 0.01 in., the open triangles. The value for CRUX in this case is 0.32×10^{7}. Though this is a more flexible arch, and should lead to less stiff response, the response was actually much stiffer than for the thicker arch. This was surprising. However, this result can be attributed to the

following: For both the thick and the thin arch a certain percentage of the strain energy is due to bending strains, and the remaining percentage is due to extensional, or membrane, strains. For the thinner arch there is very little strain energy due to bending. Unfortunately, the element displacement field is represented only by a quadratic polynomial for the extensional motion, u(x), but it uses a cubic polynomial for the out-of-plane, or bending, motion w(x). Though the element does not 'lock' in the classic sense of finite-elements, it tends toward locking behavior in the extensional mode and is thus overstiff for elements which have a higher percentage of strain energy in extensional effects than in bending effects. The thinner element thus shows this tendency to be overstiff.

To further study this locking tendency, and to minimize it, the number of elements was doubled and the results using two thickness again compared. With more elements, the stiffness of the finite-element model more closely matches the results of Huddleston. In fact, with the thicker arch, the solid squares, the results compare well. As the case of the thinner arch, the solid triangles, is stiffer, the membrane stiffening effect is again evident.

To study the effect of the number of integration intervals, and to determine if this had any influence on the stiffness of the model, the thicker arch and 16 elements were again used but with eight instead of four trapezoidal integration intervals. These results are shown as solid circles and it is evident the number of integration intervals has little influence, as the solid circles and the open squares are practically coincident.

After determining the number of elements, the number of integration intervals, and the physical parameters of the arch necessary to favorably compare the finite-element results with those of Huddleston for the intital arch response, the eigenvalue continuation method was tested. Using the arch with $h = 0.1$ and $L = 16$ which results

in CRUX = 0.32×10^{-5} and REL = 0.25, and 32 elements, the eigenvalue continuation scheme was used to branch onto the bifurcated path and to move past the limit point. The comparison of the finite-element results and Huddleston's results are shown in fig. 17. Again the values for Huddelston were obtained by interpolating form the figures in ref. 3. As illustrated in fig. 17 the finite-element results compare quite well with Huddleston's results. However, with the deep arch the bifurcated path and the symmetric path on the other side of the limit point were more difficult to follow than they had been in the shallow arch. Smaller load steps were necessary to be able to continue on the curve after an initial equilibrium solution on the adjacent path had been found using the eigenvalue continuation scheme.

FINAL COMMENTS

Presented has been a summary of a rather extensive study of a complex yet fundamental problem. The problem is complicated by the existence of multiple equilibrium solutions. A finite-element formulation with a scheme to aid in finding the multiple solutions has been developed and discussed. Comparisons with other solutions are good and lend credibility to the formulation. A users guide for the program written to implement the finite-element formulation is provided in Appendix B. At this point a variety of arches should be studied to better understand the character of the formulation, and to interpret how the formulation actually represents and interprets particular physical characteristics of arch response.

REFERENCES

1. Hyer, M.W., Johnson, E.R., and Knott, T.W., "A Study of the Response of Nonlinear Springs," Report VPI-E-89-15, August 1989.

2. Hyer, M.W., Johnson, E.R., and Knott, T.W., "A Study of the Response of Nonlinear Springs," Report VPI-E-90-01, January 1990.

3. Huddleston, J.V., "Finite Deflections and Snap-Through of High Circular Arches," J. Applied Mechanics, vol. 35, no. 4, pp. 763- 769, Dec. 1968.

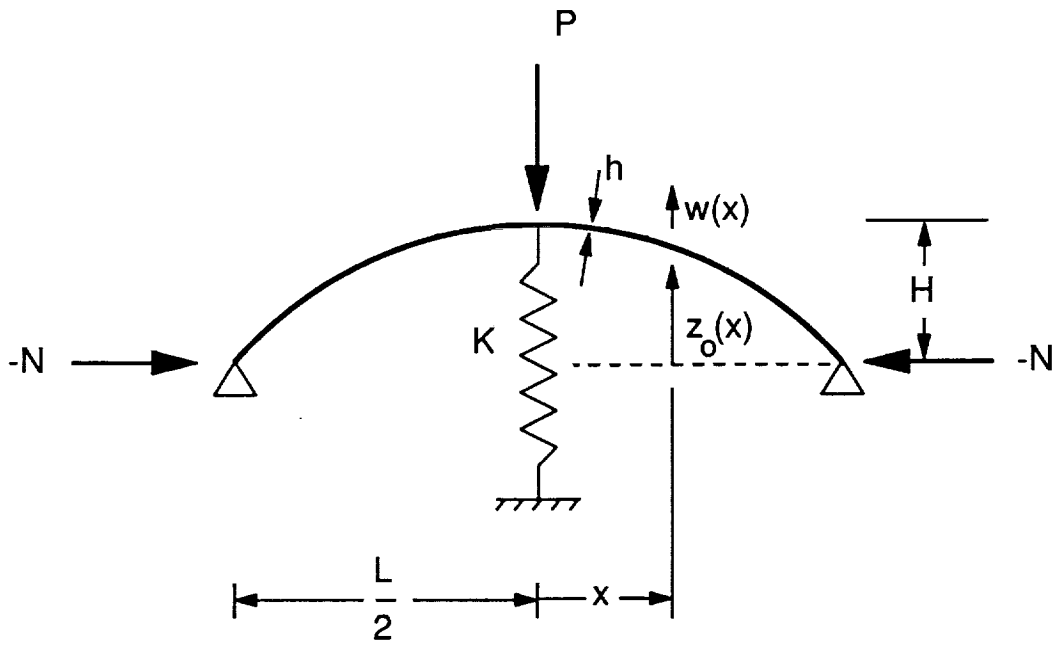

Fig. 1 - Geometry, loading, and nomenclature for shallow arch.

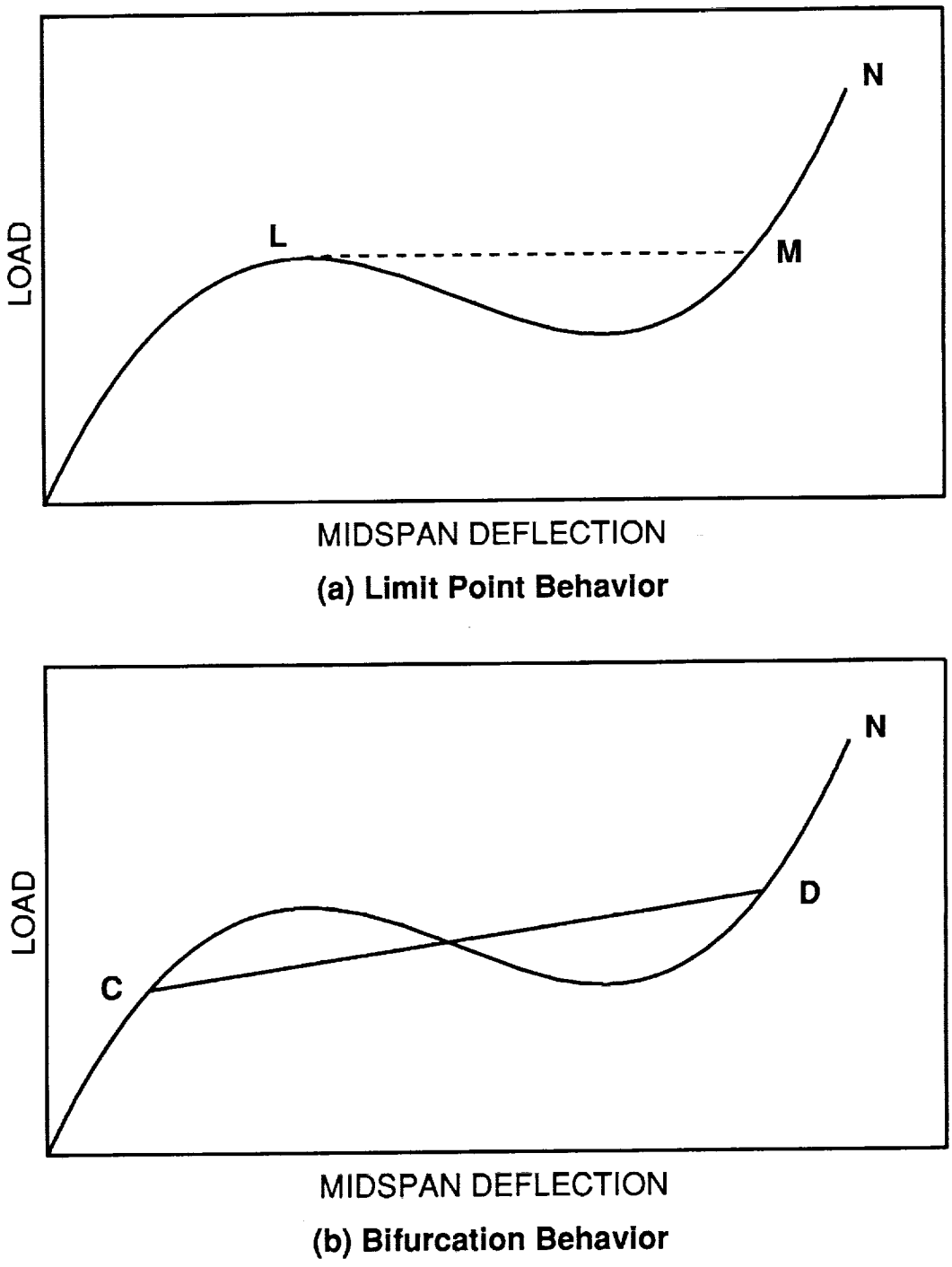

Fig. 2 - Illustration of two types of arch load-deflection behavior.

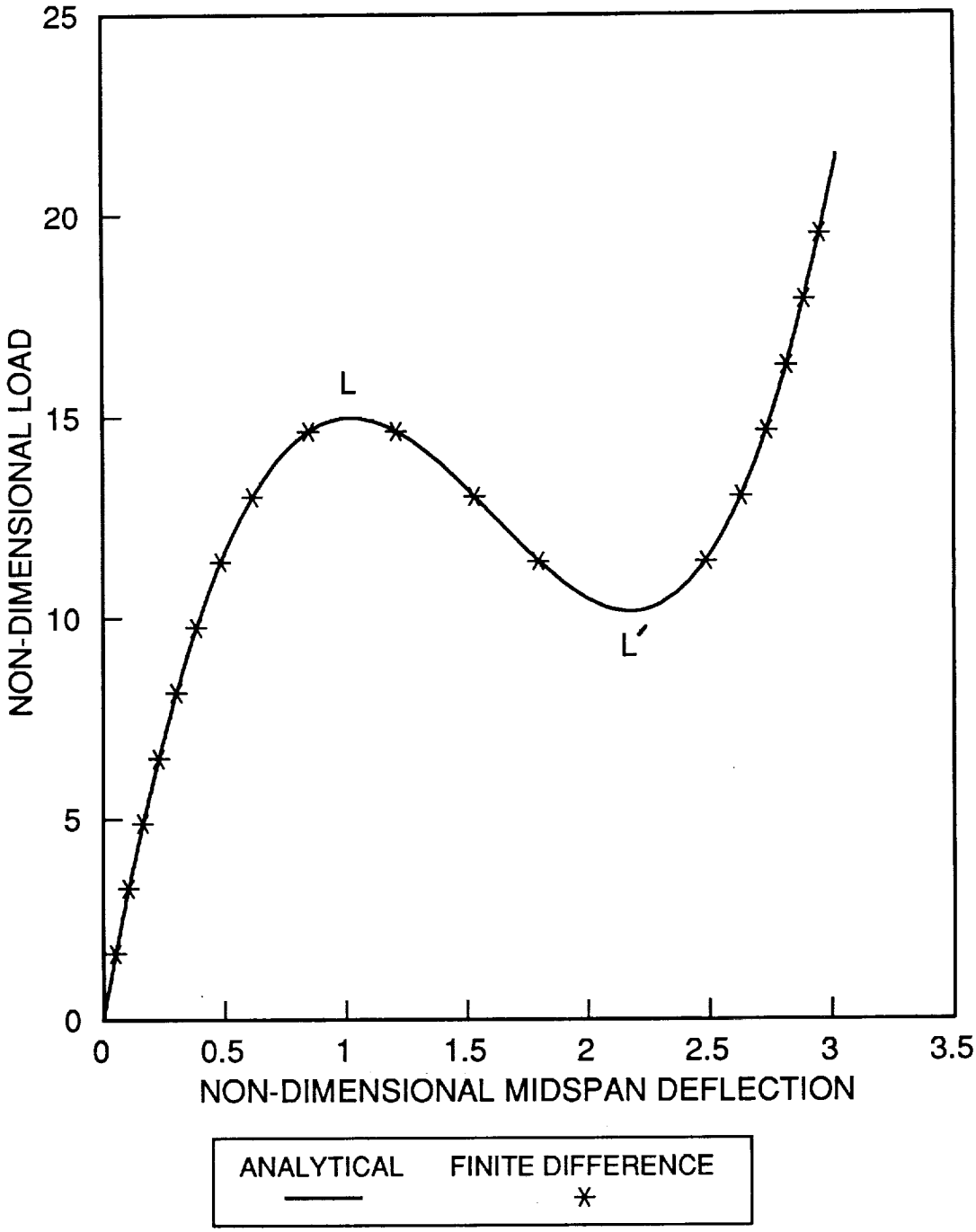

Fig. 3 - Comparison of load-deflection behavior for exact solution and finite-difference analysis, $\lambda = 1.5$, $k = 2$.

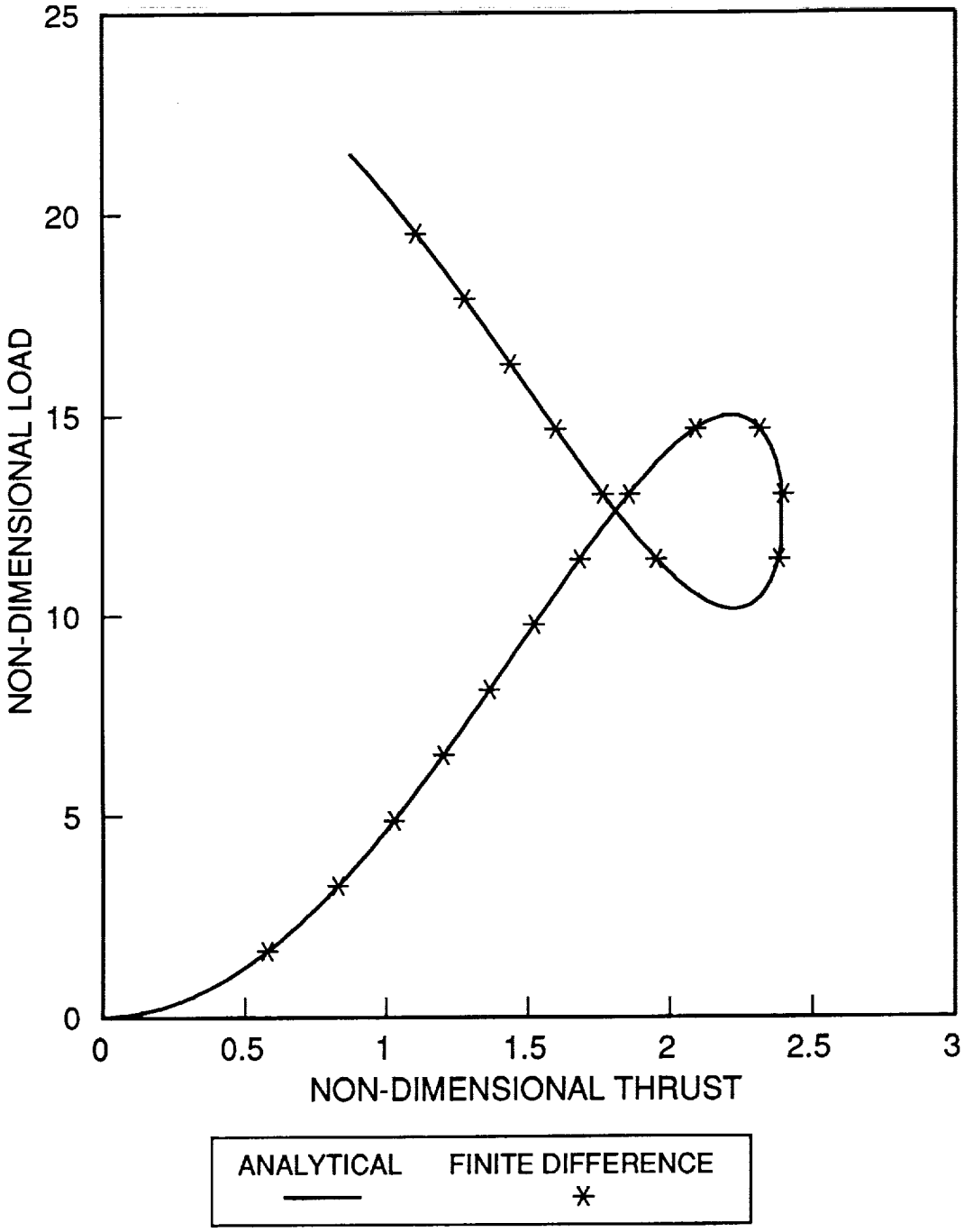

Fig. 4 - Comparison of load-thrust behavior for exact solution and finite-difference analysis, $\lambda = 1.5$, $k = 2$.

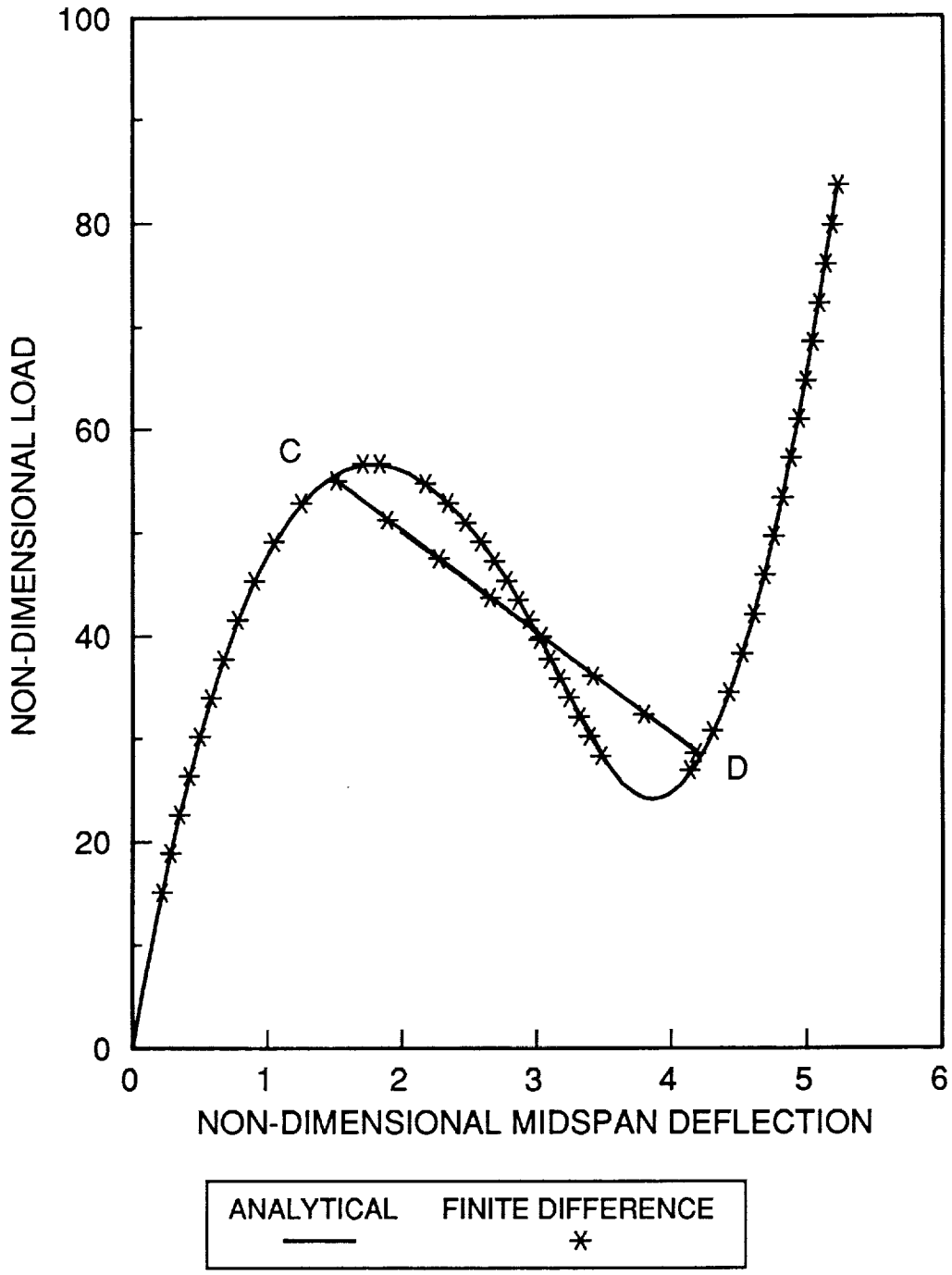

Fig. 5 - Comparison of load-deflection behavior for exact solution and finite-difference analysis, $\lambda = 2.5$, $k = 10$.

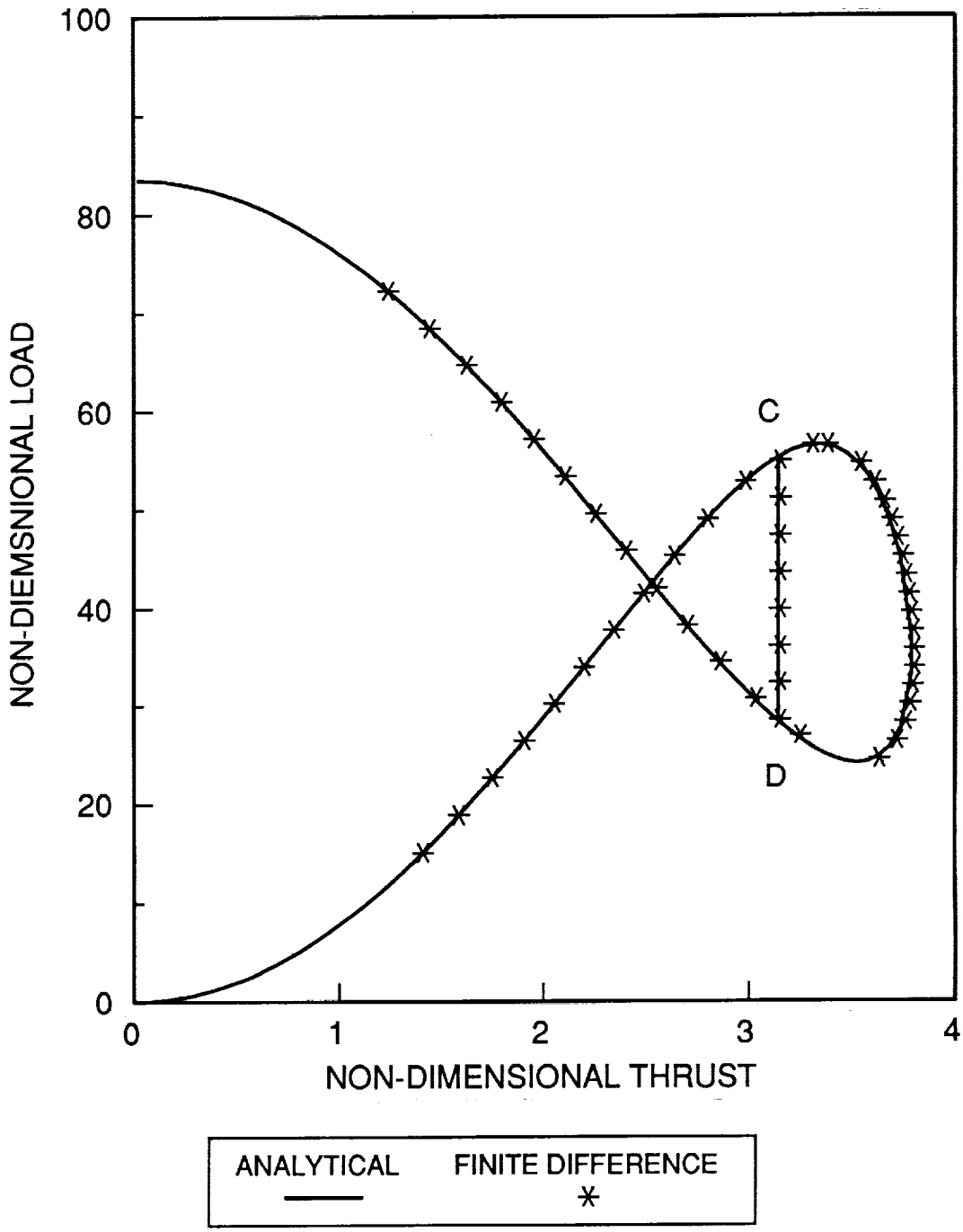

Fig. 6 - Comparison of load-thrust behavior for exact solution and finite-difference analysis, $\lambda = 2.5$, $k = 10$.

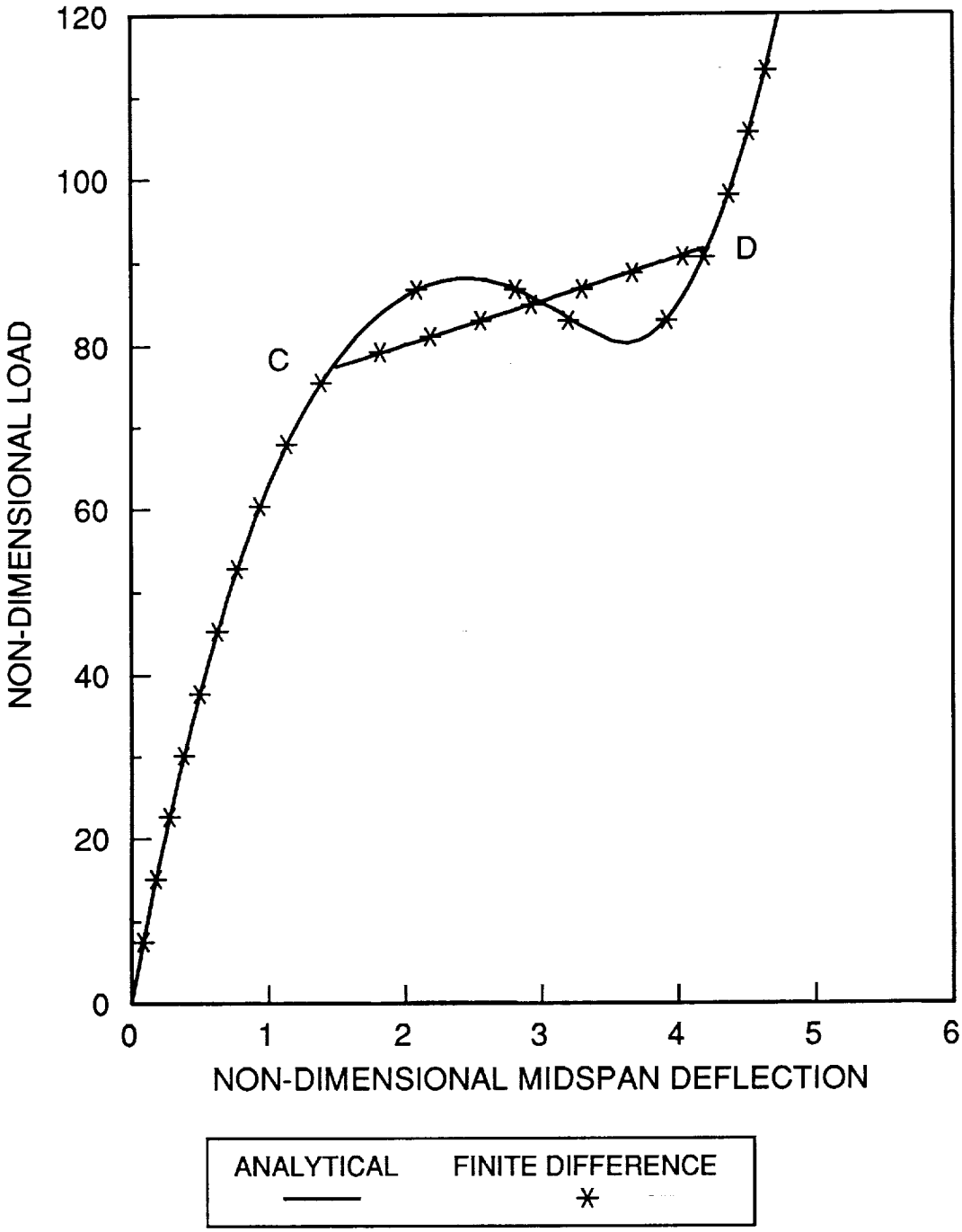

Fig. 7 - Comparison of load-deflection behavior for exact solution and finite-difference analysis, $\lambda = 2.5$, $k = 25$.

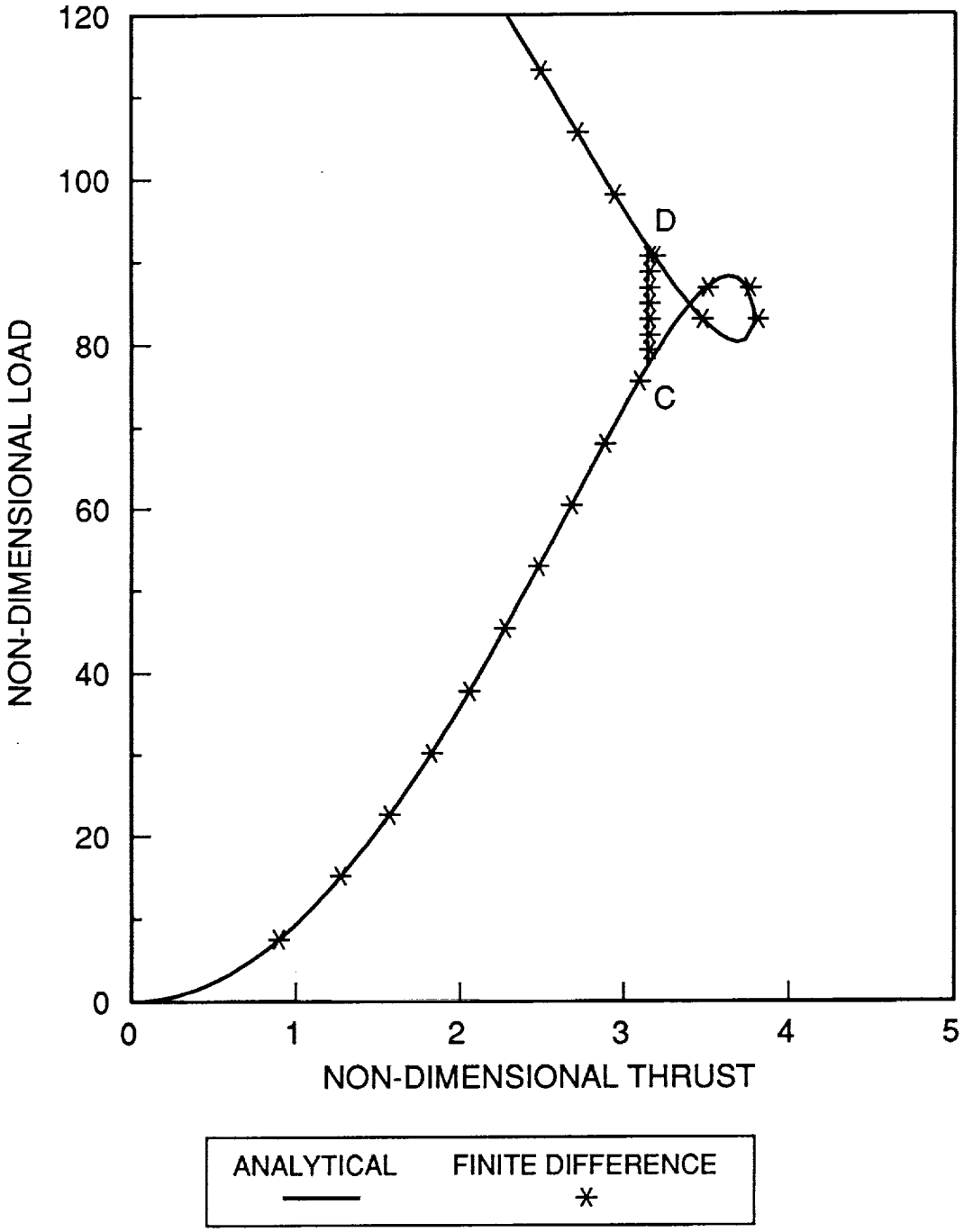

Fig. 8 - Comparison of load-thrust behavior for exact solution and finite-difference analysis, $\lambda = 2.5$, $k = 25$.

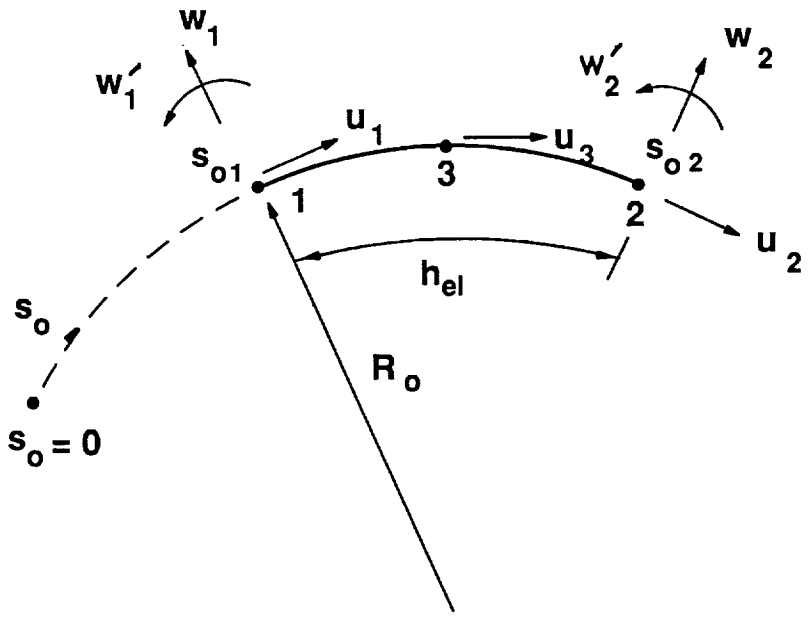

Fig. 9 - Seven degree-of-freedom arch element.

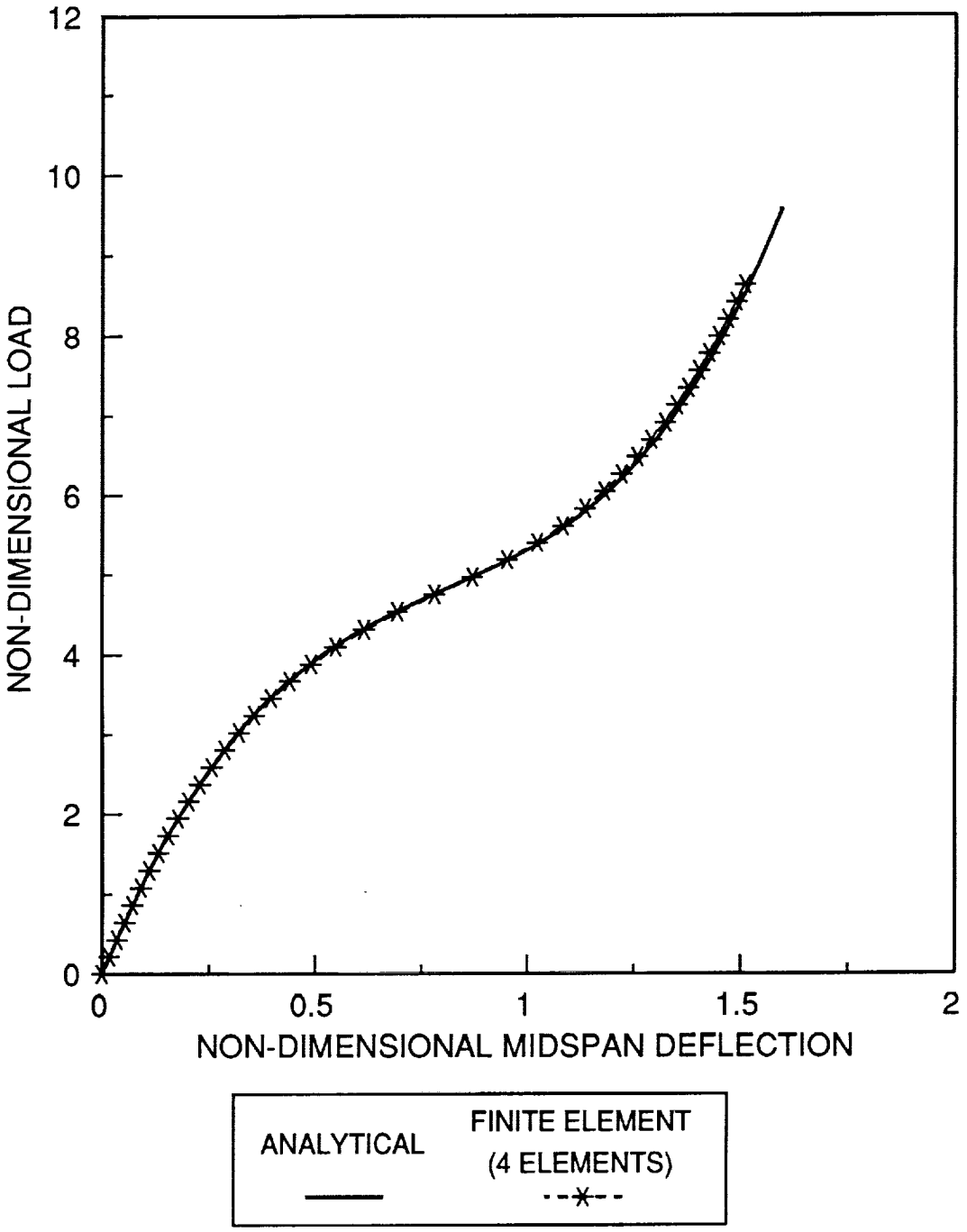

Fig. 10 - Comparison of load-deflection behavior for exact solution and finite-element analysis, $\lambda = 0.7656$, $k = 0$.

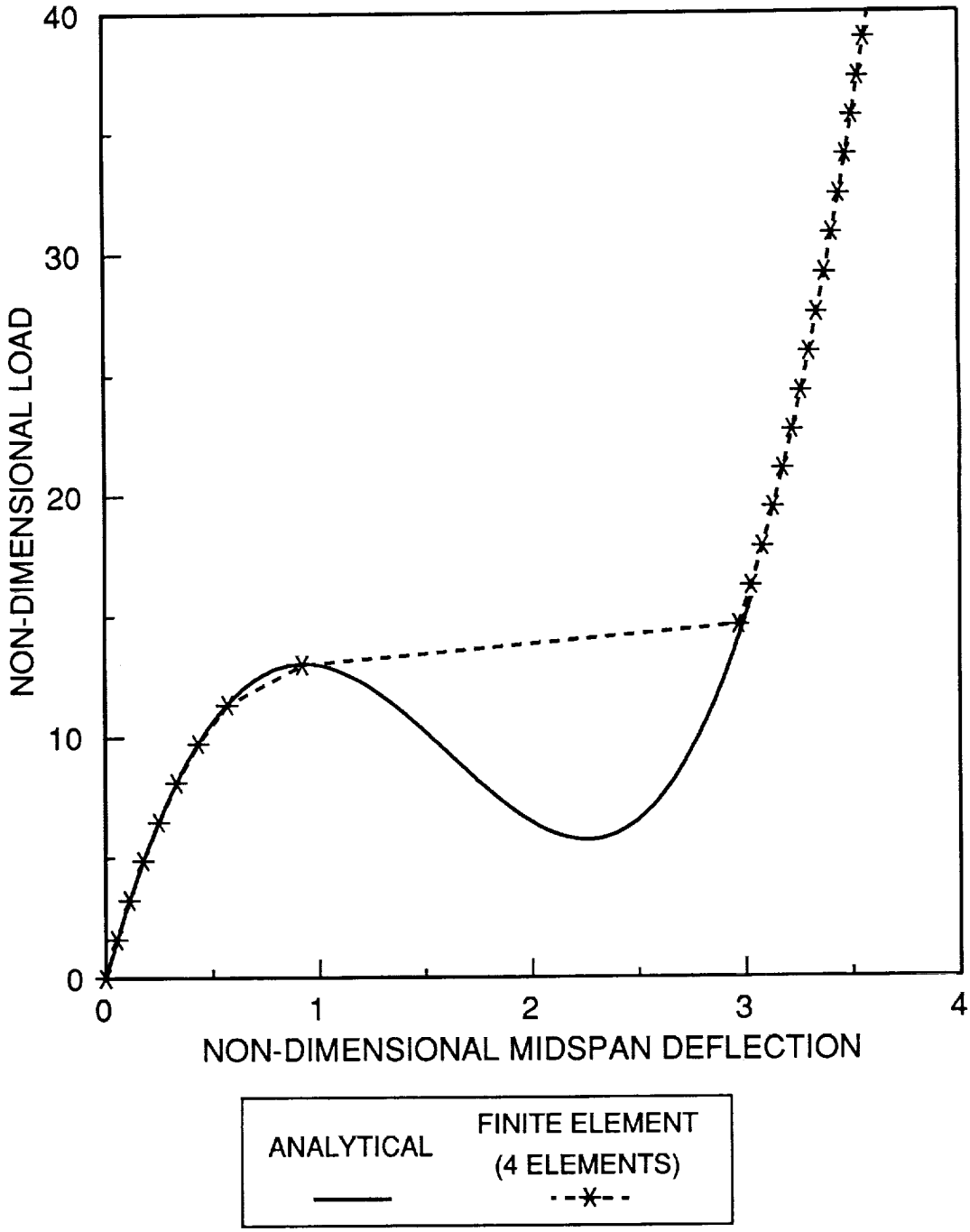

Fig. 11 - Comparison of load-deflection behavior for exact solution and finite-element analysis, $\lambda = 1.5$, $k = 0$.

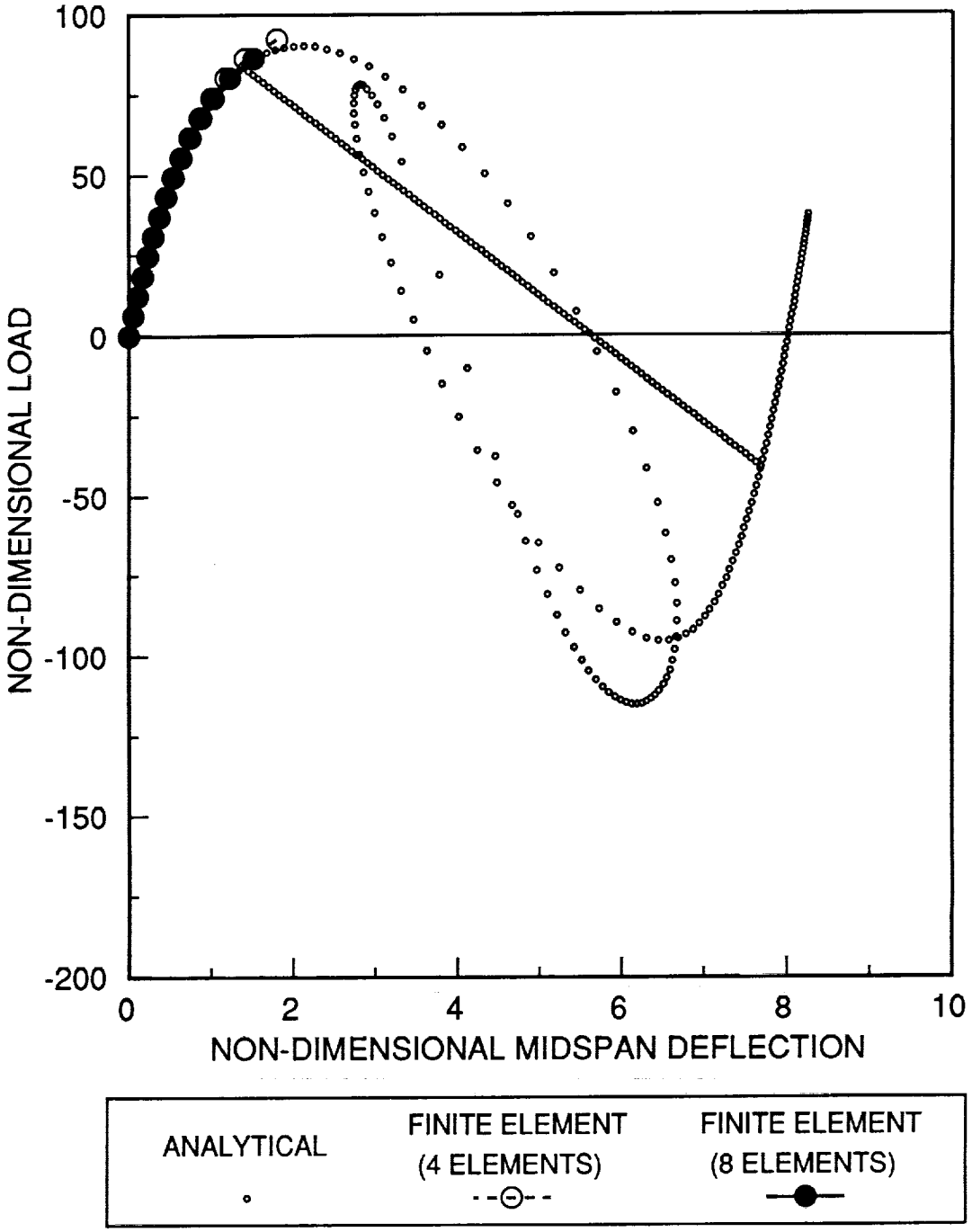

Fig. 12 - Comparison of load-deflection behavior for exact solution and finite-element analysis, $\lambda = 4$, $k = 0$.

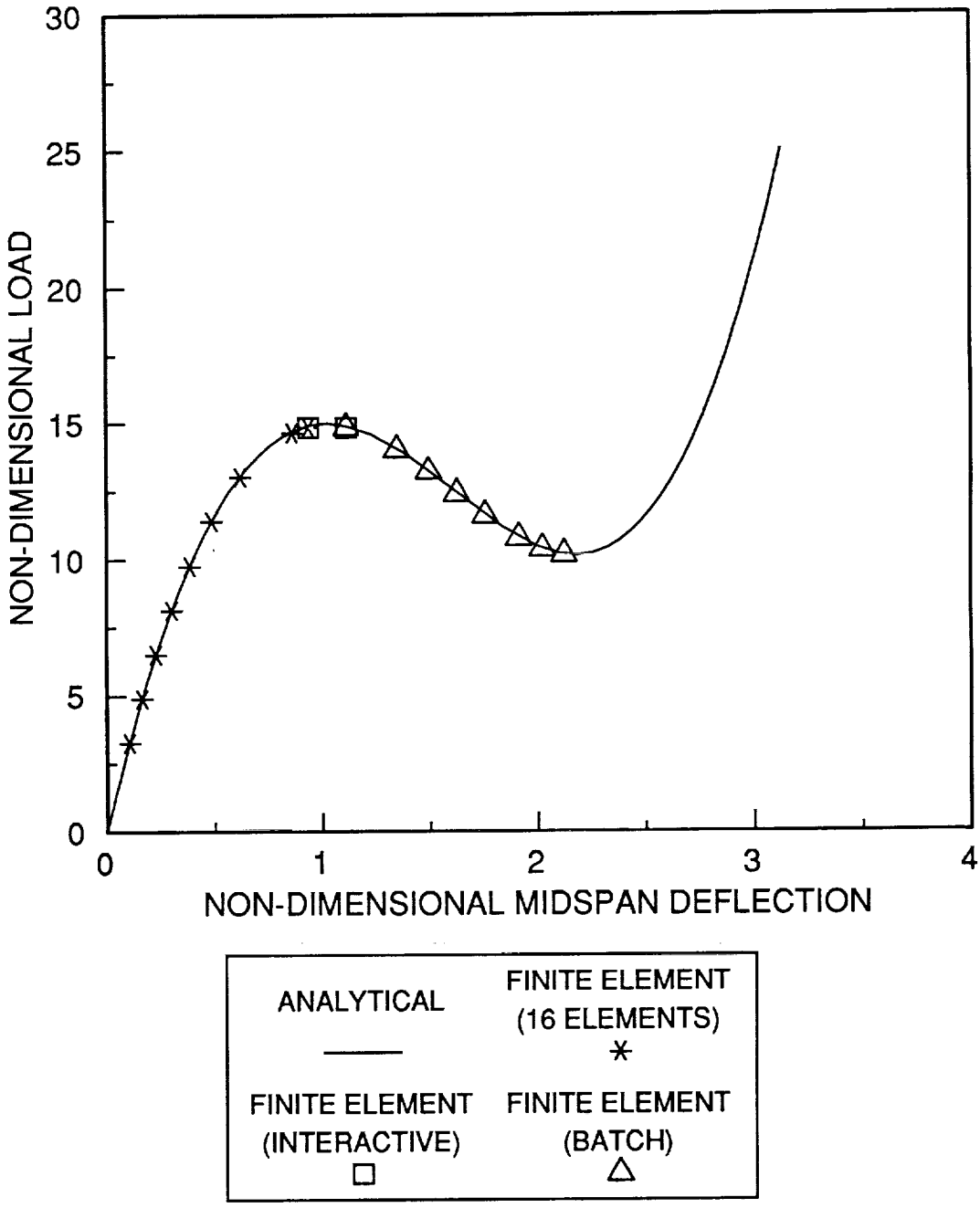

Fig. 13 - Comparison of load-deflection behavior for exact solution and finite-element analysis, $\lambda = 1.5$, $k = 2$.

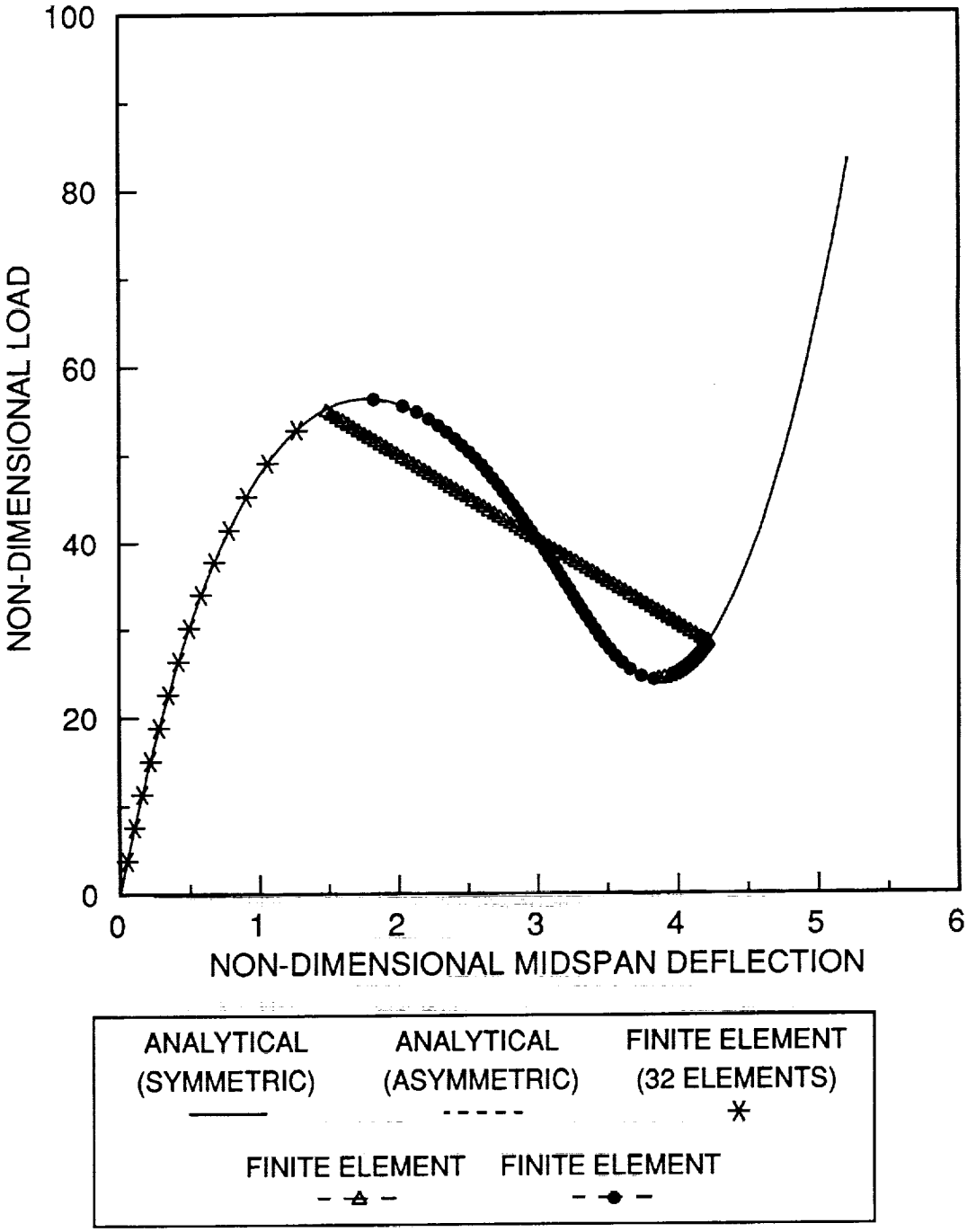

Fig. 14 - Comparison of load-deflection behavior for exact solution and finite-element analysis, $\lambda = 2.5$, $k = 10$.

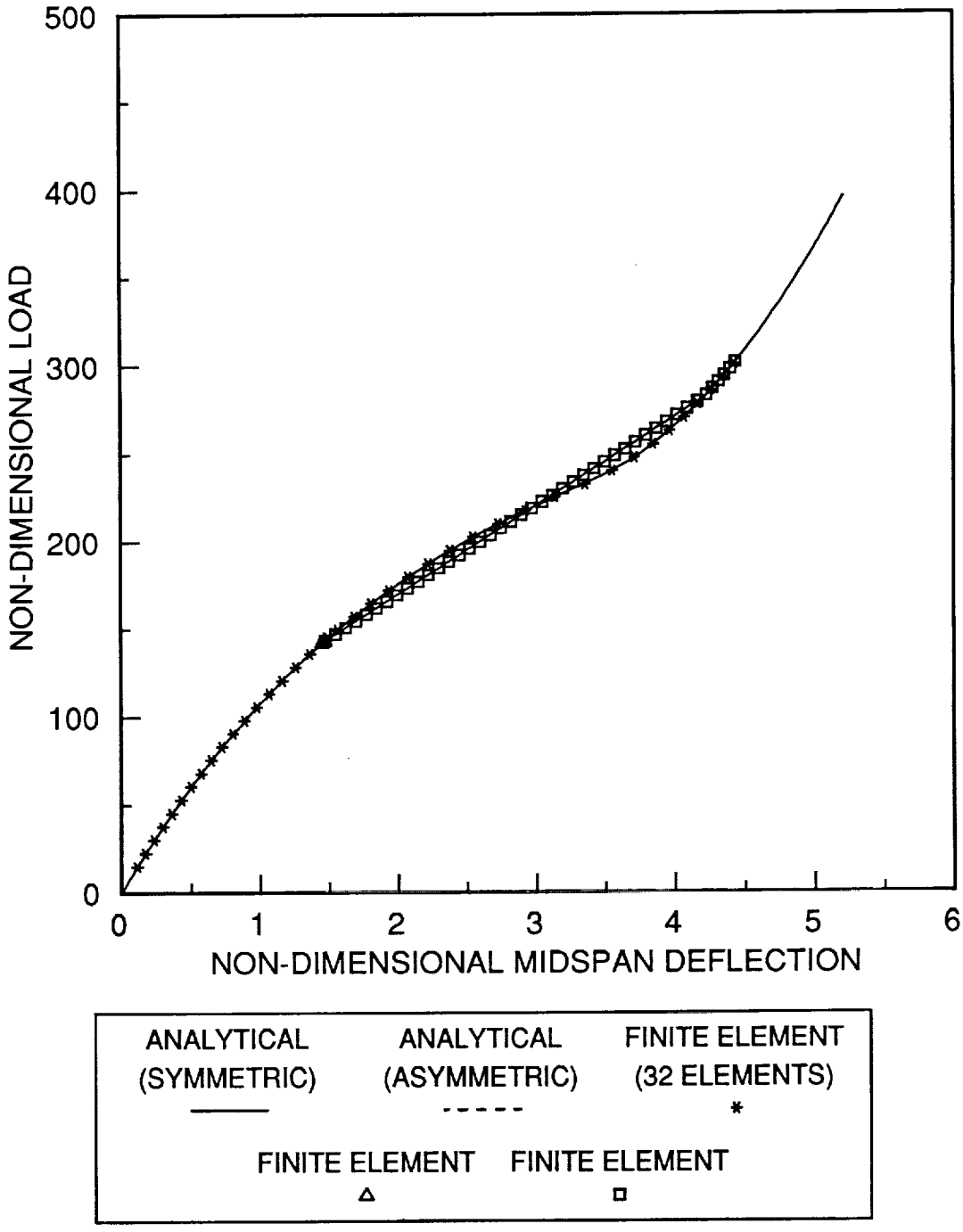

Fig. 15 - Comparison of load-deflection behavior for exact solution and finite-element analysis, $\lambda = 2.5$, $k = 70$.

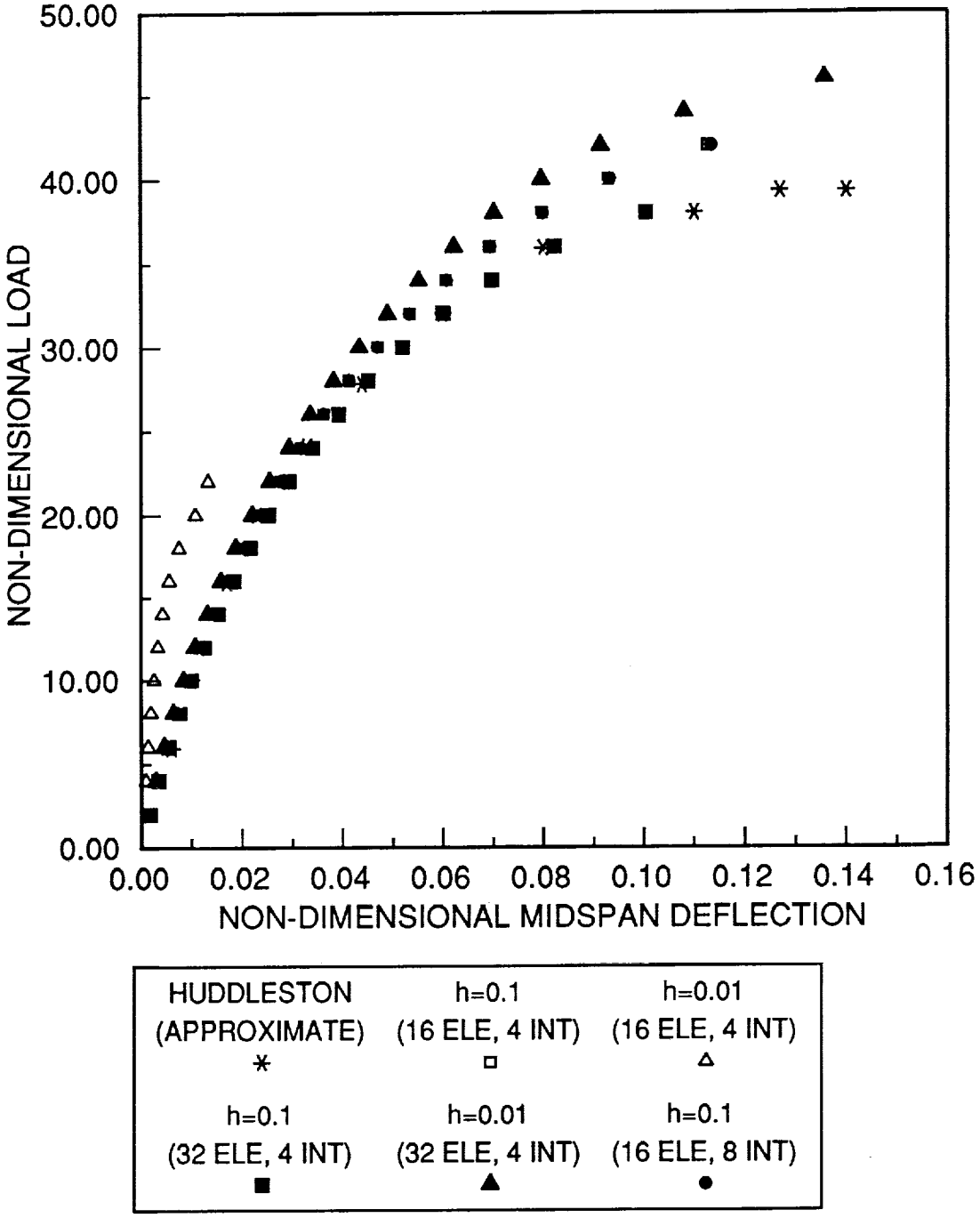

Fig. 16 - Comparison of load-deflection behavior for Huddleston's [3] results and finite-element analysis, CRUX = 0, REL = 0.25.

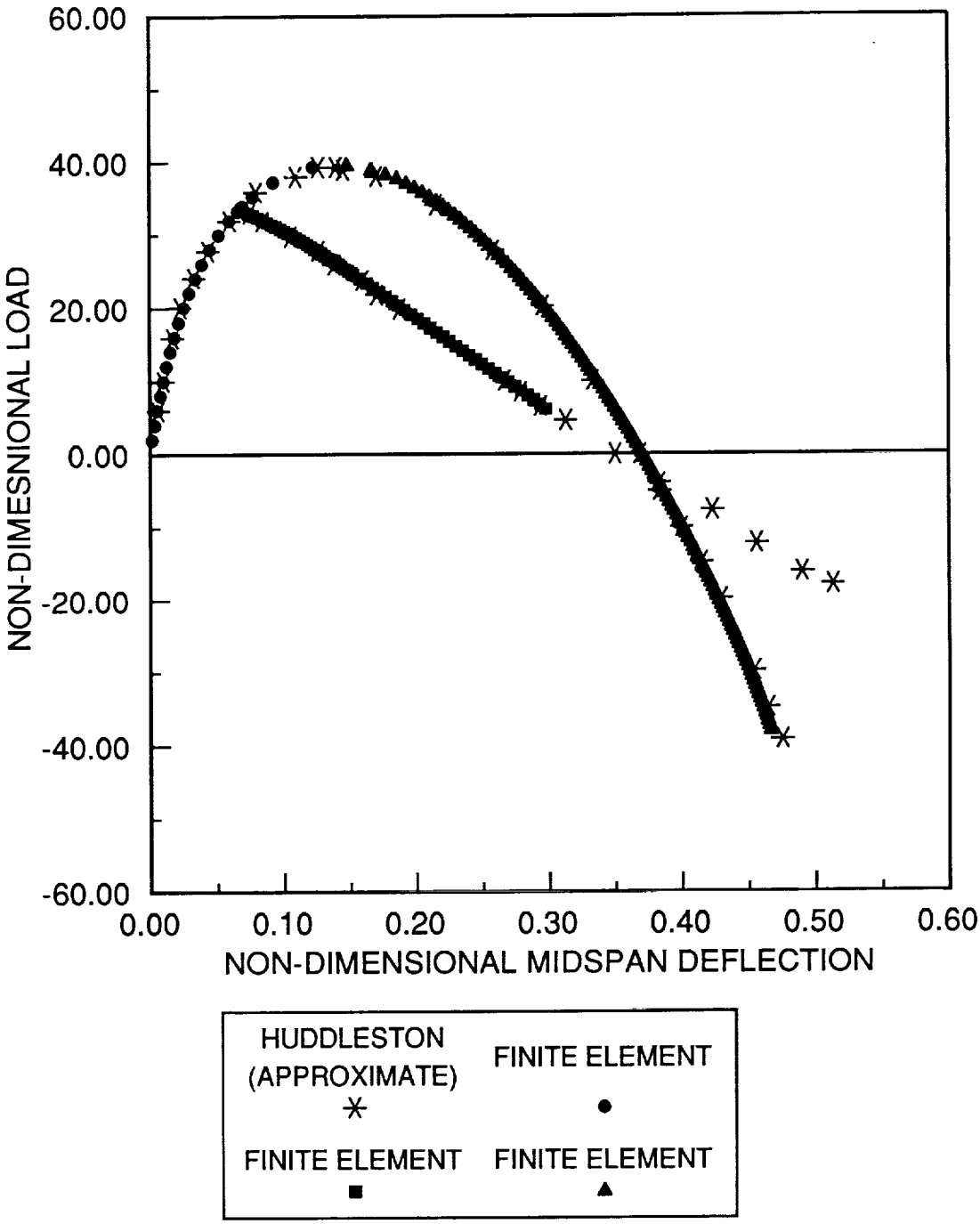

Fig. 17 - Comparison of load-deflection behavior for Huddleston's [3] results and finite-element analysis, CRUX = 0, REL = 0.25.

Appendix A

Derivation of the Governing Equations

The governing equations for the static equilibrium configurations are developed first for deep arches, and then specialized for shallow arches. We consider the large displacement response of elastic arches in the plane of their curvature.

DEEP ARCH

Kinematics

The initial, unstressed configuration of the arch is characterized by a symmetric plane curve, or reference arc, whose arc length coordinate is denoted s_o, and whose curvature in the plane, κ_o, is of one sign. Coordinate $s_o = 0$ at the left end and $s_o = S$ at the right end. The origin 0 of a right-handed cartesian system (x,y,z) is at the left end, with the x-axis passing through the end points, the y-axis perpendicular to the plane of arch, and the z-axis in the plane of the arch. See Fig. A.1. The unit vectors $\hat{\underset{\sim}{i}}$, $\hat{\underset{\sim}{j}}$ and $\hat{\underset{\sim}{k}}$ denote the positive cartesian directions x, y, and z, respectively. A point on the reference arc has a position vector $\underset{\sim}{r}_o(s_o)$ measured relative to the origin 0. Let $\hat{\underset{\sim}{T}}$ and $\hat{\underset{\sim}{N}}$ denote unit tangent and normal vectors, respectively, of the reference arc at s_o. The differential geometry of the reference arc is represented by

$$\frac{d\underset{\sim}{r}_o}{ds_o} = \hat{\underset{\sim}{T}}, \quad \frac{d\hat{\underset{\sim}{T}}}{ds_o} = \kappa_o \hat{\underset{\sim}{N}}, \quad \frac{d\hat{\underset{\sim}{N}}}{ds_o} = -\kappa_o \hat{\underset{\sim}{T}} \tag{A.1}$$

in which the unit normal $\hat{\underset{\sim}{N}}$ is directed to the concave side of the reference arc.

The material point with position $\underset{\sim}{r}_o(s_o)$ in the initial configuration occupies the point defined by position vector $\underset{\sim}{r}_o^*$ in the deformed configuration. See Fig. A.2. We introduce the displacement vector $\underset{\sim}{\Delta}_o$ such that

$$\underset{\sim}{r}_o^* = \underset{\sim}{r}_o + \underset{\sim}{\Delta}_o \tag{A.2}$$

whose components are

$$\underset{\sim}{\Delta}_o = u_o(s_o)\ \hat{\underset{\sim}{T}}(s_o) + w_o(s_o)\ \hat{\underset{\sim}{N}}(s_o)\,. \tag{A.3}$$

The differential of eq. (2) is

$$d\underset{\sim}{r}_o^* = \left(\frac{d\underset{\sim}{r}_o}{ds_o} + \frac{d\underset{\sim}{\Delta}_o}{ds_o} \right) ds_o\,. \tag{A.4}$$

Using eqs. (1) and (3), this differential is written as

$$d\underset{\sim}{r}_o^* = \left[(1 + \Gamma_T)\hat{\underset{\sim}{T}} + \Gamma_N \hat{\underset{\sim}{N}} \right] ds_o \tag{A.5}$$

where the components of the displacement gradient are

$$\begin{aligned} \Gamma_T &= u'_o - \kappa_o w_o \\ \Gamma_N &= w'_o + \kappa_o u_o \end{aligned} \tag{A.6}$$

and the prime means ordinary derivative with respect to s_o. Line element $d\underset{\sim}{r}_o$ of length ds_o in the initial configuration displaces and rotates to line element $d\underset{\sim}{r}_o^*$ of length ds_o^* in the deformed configuration. Since $(ds_o^*)^2 = d\underset{\sim}{r}_o^* \bullet d\underset{\sim}{r}_o^*$, we have from eq. (5) that

$$(ds_o^*)^2 = \left[(1 + \Gamma_T)^2 + \Gamma_N^2 \right] (ds_o)^2\,. \tag{A.7}$$

The stretch ratio λ_o is defined as ds_o^*/ds_o. Thus from eq. (7)

$$\lambda_o^2 = (ds_o^*/ds_o)^2 = (1 + \Gamma_T)^2 + \Gamma_N^2\,. \tag{A.8}$$

Green's strain ε_o of the reference arc is defined by

$$(ds_o^*)^2 - (ds_o)^2 = 2\varepsilon_o (ds_o)^2 \tag{A.9}$$

which from eq. (8) can be written as

$$\varepsilon_o = \frac{1}{2}\left(\lambda_o^2 - 1\right) \tag{A.10}$$

or

$$\varepsilon_o = \Gamma_T + \frac{1}{2}\Gamma_T^2 + \frac{1}{2}\Gamma_N^2 . \tag{A.11}$$

Similar to eqs. (1) for the initial state, the differential geometry of reference arc in the deformed state is given by

$$\frac{d\underset{\sim}{r}_o^*}{ds_o^*} = \hat{\underset{\sim}{t}}\,, \quad \frac{d\hat{\underset{\sim}{t}}}{ds_o^*} = \kappa_o^* \hat{\underset{\sim}{n}}\,, \quad \frac{d\hat{\underset{\sim}{n}}}{ds_o^*} = -\kappa_o^* \hat{\underset{\sim}{t}} \tag{A.12}$$

in which κ_o^* is the curvature at s_o^*. From the first of eqs. (12), and eqs. (5) and (8), the unit tangent vector is

$$\hat{\underset{\sim}{t}} = \frac{1}{\lambda_o}(1 + \Gamma_T)\,\hat{\underset{\sim}{T}} + \frac{1}{\lambda_o}\Gamma_N\,\hat{\underset{\sim}{N}} . \tag{A.13}$$

We define the rotation of line element $d\underset{\sim}{r}_o$ in the initial state into $d\underset{\sim}{r}_o^*$ in the deformed state by the angle β. Angle β is measured positive clockwise from $\hat{\underset{\sim}{T}}$ to $\hat{\underset{\sim}{t}}$ such that

$$\hat{\underset{\sim}{t}} = \cos\beta\,\hat{\underset{\sim}{T}} + \sin\beta\,\hat{\underset{\sim}{N}} . \tag{A.14}$$

Comparing eqs. (13) and (14) gives the trigonometric functions of the angle β in terms of the components of the displacement gradients as

$$\lambda_o \cos\beta = 1 + \Gamma_T \tag{A.15}$$

$$\lambda_o \sin\beta = \Gamma_N . \tag{A.16}$$

Taking the derivative of $\hat{\underset{\sim}{t}}$ in eq. (14) with respect to s_o^*, using the chain rule and eqs. (1) and (8), and comparing this result to the second of eqs. (12), we find that

$$\lambda_o \kappa_o^* = \beta' + \kappa_o \tag{A.17}$$

$$\hat{\underset{\sim}{n}} = -\sin\beta\ \hat{\underset{\sim}{T}} + \cos\beta\ \hat{\underset{\sim}{N}} \tag{A.18}$$

where $\hat{\underset{\sim}{n}}$ is the unit normal to the reference arc in the deformed state at s_o^*. The unit vector $\hat{\underset{\sim}{n}}$ is rotated 90° clockwise with respect to the tangent vector $\hat{\underset{\sim}{t}}$.

A parallel arc to the reference arc in the initial state is defined by the position vector

$$\underset{\sim}{r} = \underset{\sim}{r}_o + \zeta \hat{\underset{\sim}{N}} \tag{A.19}$$

in which ζ is the coordinate along the normal at s_o. Coordinate ζ is constant for a parallel arc. Using eqs. (1) the differential line element tangent to the parallel arc is

$$d\underset{\sim}{r} = (1 - \zeta\kappa_o) ds_o\ \hat{\underset{\sim}{T}}, \quad \zeta = \text{constant}. \tag{A.20}$$

The magnitude of eq. (20) is

$$ds = (1 - \zeta\kappa_o) ds_o \tag{A.21}$$

in which ds is the arc length of the parallel arc. A material point located by position vector $\underset{\sim}{r}$ in the initial state is located in the deformed state by position vector $\underset{\sim}{r}^*$. The displacement of this material point on the parallel arc is given by the vector difference $\underset{\sim}{r}^* - \underset{\sim}{r}$, and we invoke the Kirchhoff-Love hypothesis to relate this displacement to the displacement of a material point on the reference arc. The Kirchhoff-Love hypothesis is

$$\underset{\sim}{r}^* - \underset{\sim}{r} = \underset{\sim}{\Delta}_o + \zeta\left(\hat{\underset{\sim}{n}} - \hat{\underset{\sim}{N}}\right). \tag{A.22}$$

Using eqs. (19) and (2) this becomes

$$\underset{\sim}{r}^{*} = \underset{\sim}{r}_{o}^{*} + \zeta \hat{\underset{\sim}{n}} . \tag{A.23}$$

The differential of $\underset{\sim}{r}^{*}$ in eq. (23) is

$$d\underset{\sim}{r}^{*} = d\underset{\sim}{r}_{o}^{*} + \zeta d\hat{\underset{\sim}{n}} \tag{A.24}$$

which from the relations in eqs. (12) may be written as

$$d\underset{\sim}{r}^{*} = (1 - \zeta\kappa_{o}^{*})ds_{o}^{*}\, \hat{\underset{\sim}{t}} . \tag{A.25}$$

Thus, the magnitude of eq. (25) is the differential arc length on the parallel arc in the deformed state; i.e.,

$$ds^{*} = (1 - \zeta\kappa_{o}^{*})ds_{o}^{*} . \tag{A.26}$$

The stretch ratio for the parallel arc is defined as

$$\lambda = ds^{*}/ds \tag{A.27}$$

which from eqs. (21) and (26) becomes

$$\lambda = \lambda_{o}(1 - \zeta\kappa_{o}^{*})/(1 - \zeta\kappa_{o}) . \tag{A.28}$$

Finally, the Green's strain for a parallel arc is

$$\varepsilon = \frac{1}{2}(\lambda^{2} - 1) \tag{A.29}$$

or in view of eq. (28) and (10) this becomes

$$\varepsilon = \frac{1}{2}\left[(1 + 2\varepsilon_{o})\left(\frac{1 - \zeta\kappa_{o}^{*}}{1 - \zeta\kappa_{o}}\right)^{2} - 1\right] . \tag{A.30}$$

Equilibrium

Consider equilibrium of an infinitesimal element of the arch in the deformed state. The internal actions are a force vector $\underset{\sim}{F}^*$ in the plane and a moment vector $M^*\hat{\underset{\sim}{j}}$. Moment M^* is positive clockwise acting on a positive s_o^* face. Static equilibrium in the limit as $ds_o^* \to 0$ requires

$$\frac{d\underset{\sim}{F}^*}{ds_o^*} = 0$$
$$\frac{dM^*}{ds_o^*}\hat{\underset{\sim}{j}} + \hat{\underset{\sim}{t}} \times \underset{\sim}{F}^* = 0 . \tag{A.31}$$

At the point $s_o^* = s_m^*$, where the spring is attached and the applied downward load P acts (Fig. A.2), eqs. (31) are invalid and they are replaced by the transition equilibrium conditions. These transition equilibrium equations are

$$\left[\underset{\sim}{F}^*(s_m^*)\right] = P\hat{\underset{\sim}{k}} + F_s\hat{\underset{\sim}{u}}$$
$$\left[M^*(s_m^*)\right] = 0 \tag{A.32}$$

in which the $[\mathbb{Q}]$ means the difference between the quantity $\mathbb{Q}$ in the brackets evaluated on the right side of s_m^* and the quantity evaluated on the left side of s_m^*. The spring force vector acting on the arch is $-F_s\hat{\underset{\sim}{u}}$, where

$$F_s = K(\ell - \ell_o) \tag{A.33}$$

and

$$\ell\hat{\underset{\sim}{u}} = \ell_o\hat{\underset{\sim}{k}} + \underset{\sim}{\Delta}_o(s_m) . \tag{A.34}$$

In eqs. (33) and (34) ℓ_o and ℓ represent the unstretched and stretched spring lengths, K is the spring stiffness, and $\underset{\sim}{\Delta}_o(s_m)$ is the displacement vector of the arch at the spring attachment point (midspan). The magnitude of the vector eq. (34) is

$$\ell^2 = (\ell_o - w_m)^2 + u_m^2 \tag{A.35}$$

in which $\underset{\sim}{\Delta}_o(s_m) = w_m\, \hat{\underset{\sim}{N}}(s_m) + u_m\, \hat{\underset{\sim}{T}}(s_m)$, $\hat{\underset{\sim}{k}} \bullet \hat{N}(s_m) = -1$, and $\hat{\underset{\sim}{i}} \bullet \hat{\underset{\sim}{T}}(s_m) = 1$ for the midspan location on the symmetric arch. The unit vector $\hat{\underset{\sim}{u}}$ is directed from the base of the spring along its line of action toward its connection to the arch in the deformed state. From eq. (34)

$$\hat{\underset{\sim}{u}} = \frac{\ell_o - w_m}{\ell}\, \hat{\underset{\sim}{k}} + \frac{u_m}{\ell}\, \hat{\underset{\sim}{i}}\,. \tag{A.36}$$

Virtual Work

Consider an infinitesimal virtual displacement of the arch from its equilibrium configuration in the deformed state. The increment in the virtual work of the external loads (P and F_s) is

$$\delta W_{ext} = \left[P\left(-\hat{\underset{\sim}{k}}\right) + F_s\left(-\hat{\underset{\sim}{u}}\right) \right] \bullet \delta \underset{\sim}{r}_o^*(s_m^*) \tag{A.37}$$

in which $\delta \underset{\sim}{r}_o^*$ denotes a kinematically admissible virtual change in position. Using the first eqs. (32) this expression becomes

$$\delta W_{ext} = -\left[\underset{\sim}{F}^*(s_m^*) \right] \bullet \delta \underset{\sim}{r}_o^*(s_m^*)\,. \tag{A.38}$$

Equation (38) is equivalent to

$$\delta W_{ext} = \int_0^{s_m^*} + \int_{s_m^*}^{S^*} \underset{\sim}{F}^* \bullet \frac{d(\delta \underset{\sim}{r}_o^*)}{ds_o^*}\, ds_o^*\,, \tag{A.39}$$

since the ends of the arch are immovable and the internal force vector is spatially constant on the open intervals $0 < s_o^* < s_m^*$ and $s_m^* < s_o^* < S^*$ via eq. (30). Immovable end points imply the kinematically admissible variations must satisfy

$$\delta \underset{\sim}{r}_o^*(0) = \delta \underset{\sim}{r}_o^*(S^*) = 0 . \tag{A.40}$$

The virtual displacement is performed following a material point which is identified by a fixed value of s_o in the initial state. Thus, $\delta \underset{\sim}{r}_o^*$ implies a change in position (displacement) holding s_o fixed. As a result, the variational operator "δ" and ordinary derivative $d(\ )/ds_o$ commute, but "δ" and $d(\ )/ds_o^*$ do not. Using the chain rule a couple of times, the spatial derivative of the variation in the integrand of eq. (39) may be manipulated as follows

$$\begin{aligned}
\frac{d}{ds_o^*}\left(\delta \underset{\sim}{r}_o^*\right) &= \frac{d}{ds_o}\left(\delta \underset{\sim}{r}_o^*\right)\frac{ds_o}{ds_o^*} \\
&= \delta\left(\frac{d\underset{\sim}{r}_o^*}{ds_o}\right)\frac{1}{\lambda_o} \\
&= \delta\left(\lambda_o \hat{\underset{\sim}{t}}\right)\frac{1}{\lambda_o} \\
&= \frac{\delta\lambda_o}{\lambda_o}\hat{\underset{\sim}{t}} + \delta\hat{\underset{\sim}{t}} \\
&= \frac{\delta\lambda_o}{\lambda_o}\hat{\underset{\sim}{t}} + \delta\beta\,\hat{\underset{\sim}{n}} .
\end{aligned} \tag{A.41}$$

In the second step of the above manipulations eqs. (8) and the first of eqs. (12) were used to rewrite the spatial derivative, and in the last step the variation of the unit tangent vector was obtained from eq. (14) recognizing that unit vectors $\hat{\underset{\sim}{T}}(s_o)$ and $\hat{\underset{\sim}{N}}(s_o)$ do not vary since s_o is fixed. Substitute eq. (41) into (39) to get

$$\delta W_{ext} = \int_0^{s_m^*} + \int_{s_m^*}^{S^*}\left[F_t^*\frac{\delta\lambda_o}{\lambda_o} + F_n^*\delta\beta\right]ds_o^* \tag{A.42}$$

in which F_t^* and F_n^* are the tangent and normal components of $\underset{\sim}{F}$ in the deformed state. The second of equilibrium eqs. (31) can be used to eliminate F_n^* in (42) in terms of the derivative of the moment. That is,

$$\delta W_{ext} = \int_0^{s_m^*} + \int_{s_m^*}^{S^*} \left[F_t \frac{\delta \lambda_o}{\lambda_o} - \frac{dM^*}{ds_o^*} \delta\beta \right] ds_o^* . \tag{A.43}$$

The following identity is valid for simply supported ends ($M^* = 0$) or for prescribed end rotations (β) (also note M^* and β are continuous at the point load)

$$\int_0^{S^*} \frac{d}{ds_o^*} \left(M^* \delta\beta \right) \, ds_o^* = 0 .$$

Expanding

$$\begin{aligned} \int_0^{S^*} \frac{dM^*}{ds_o^*} \delta\beta \, ds_o^* &= - \int_0^{S^*} M^* \frac{d}{ds_o^*} (\delta\beta) ds_o^* \\ &= - \int_0^{S^*} M^* \delta\beta' \frac{1}{\lambda_o} \, ds_o^* . \end{aligned} \tag{A.44}$$

Substituting eq. (44) into (43) and changing the integration variable from s_o^* to s_o, we get

$$\delta W_{ext} = \int_0^{s_m} + \int_{s_m}^{S} \left[F_t^* \delta\lambda_o + M^* \delta\beta' \right] ds_o . \tag{A.45}$$

The right hand side of eq. (45) is the internal virtual work, i.e.

$$\delta W_{int} = \int_0^{s_m} + \int_{s_m}^{S} \left[F_t^* \delta\lambda_o + M^* \delta\beta' \right] ds_o . \tag{A.46}$$

By eq. (45) we have shown, for an arch in equilibrium, that the internal virtual work is equal to the external virtual work for every kinematically admissible variation of the displacements with respect to the actual equilibrium displacements. In the finite el-

ement solution we use the converse of this statement; i.e., we impose $\delta W_{ext} = \delta W_{int}$ for every kinematically admissible variation to ensure equilibrium of the arch.

From eq. (46) the internal virtual work of the arch in the deformed state per unit arc length of the initial state is

$$\delta W'_{int} = F_t^* \delta\lambda_o + M^* \delta\beta' . \tag{A.47}$$

The force and moment components in eq. (47) are defined in terms of the normal (Cauchy) stress component τ_{ss} in the deformed state by

$$(F_t^* , M^*) = \int_A \tau_{ss}(1 , -\zeta) dA \tag{A.48}$$

in which A is the cross-sectional area of the arch. The cross-sectional area does not change from its value in the initial state in the theory. Substituting eq. (48) into (47) results in

$$\delta W'_{int} = \int_A \tau_{ss}(\delta\lambda_o - \zeta\delta\beta') \, dA \tag{A.49}$$

which can be rewritten in terms of the variation of the parallel arc stretch ratio (see eqs. (28) and (17)) as

$$\delta W'_{int} = \int_A \tau_{ss}\delta\lambda(1 - \zeta\kappa_o) \, dA . \tag{A.50}$$

Division and multiplication by λ in the integrand of eq. (50) gives

$$\delta W'_{int} = \int_A (\tau_{ss}/\lambda) \, \lambda\delta\lambda \, (1 - \zeta\kappa_o) \, dA . \tag{A.51}$$

The result in eq. (51) is written as

$$\delta W'_{int} = \int_A S_{ss}\delta\varepsilon\ (1 - \zeta\kappa_o)\ dA \tag{A.52}$$

in which $S_{ss}(= \tau_{ss}/\lambda)$ is the second Piola-Kirchhoff normal stress defined in the initial state, and $\delta\varepsilon$ is the first variation of Green's strain for a parallel arc; see eq. (29). The second-Piola Kirchhoff stress tensor and Green's strain tensor are the conjugate variables frequently used for finite element analysis of finite deformation in solid mechanics.

Approximation for Thin Arches and Small Strains

The Green's strain ε is a quadratic function of the thickness coordinate and curvatures as shown by eq. (30). For thin arches $|\zeta\kappa_o| \leq h\kappa_o$, where h is the arch thickness, and $h\kappa_o$ is small with respect to unity. We approximate the strain of a parallel arc for a thin arch as follows: Multiply eq. (30) by the factor $(1 - \zeta\kappa_o)$ and expand the right hand side of this result in a power series in ζ to get

$$(1 - \zeta\kappa_o)\varepsilon = \varepsilon_o - \zeta\left[(1 + 2\varepsilon_o)\dot{\kappa_o} - (1 + \varepsilon_o)\kappa_o\right] + 0(\zeta^2)\,. \tag{A.53}$$

Now we assume that the strain ε_o of the reference arc is small with respect to unity, and neglect terms quadratic in ζ in eq. (53), to get

$$\varepsilon \simeq \frac{\varepsilon_o - \zeta(\dot{\kappa_o} - \kappa_o)}{(1 - \zeta\kappa_o)}\,. \tag{A.54}$$

Equation (54) is used to approximate the small strain behavior of thin arches.

The approximation to the strain in eq. (54) is substituted into the expression (52) for the internal virtual work per unit initial arc length to obtain

$$\delta W'_{int} = N\delta\varepsilon_o + M\delta\dot{\kappa_o} \tag{A.55}$$

in which the second Piola-Kirchhoff stress resultants are

$$(N,M) = \int_A S_{ss}(1, \quad -\zeta)\ dA\,. \tag{A.56}$$

Equation (55) shows that N and ε_o, and M and $(\kappa_o^* - \kappa_o)$, are conjugate variables. The Green's strain for the reference arc is given in terms of the displacement gradients in eq. (11). The change in reference arc curvature $\kappa_o^* - \kappa_o$ is determined from eqs. (15), (16) and (17). Consistent with the assumption of small strains, the stretch ratio $\lambda_o \sim 1$, since $\lambda_o^2 = 1 + 2\varepsilon_o$. Thus for small strains eq. (17) determines the change in curvature as

$$\kappa_o^* - \kappa_o \simeq \beta'\,. \tag{A.57}$$

The expression for the rotation gradient in terms of the displacement gradients Γ_T and Γ_N is determined by differentiating eqs. (15) and (16) with respect to s_o, and then solving the resulting two simultaneous equations in $\lambda_o\beta'$ and λ'_o. The result of this procedure is

$$\begin{aligned} \lambda_o\beta' &= (\cos\beta)\ \Gamma'_N - (\sin\beta)\ \Gamma'_T \\ \lambda'_o &= (\sin\beta)\ \Gamma'_N + (\cos\beta)\ \Gamma'_T\,. \end{aligned} \tag{A.58}$$

In regards to computing the rotation β (see eqs. (15) and (16)) and the rotation gradient β', under the assumption of small strains, it is reasonable to consider an inextensional theory. That is, λ_o is identically equal to unity for all displacements. This inextensional constraint implies $\lambda'_o = 0$ in the second of eqs. (58) and $\lambda_o = 1$ in the first equation. If the resulting system is used to eliminate Γ'_T, then the rotation gradient is

$$\beta' = \frac{\Gamma'_N}{\cos\beta} = \frac{\Gamma'_N}{\sqrt{1 - \Gamma_N^2}} \tag{A.59}$$

since eqs. (15) and (16) imply

$$\cos \beta = 1 + \Gamma_T = \sqrt{1 - \Gamma_N^2} \tag{A.60}$$

$$\sin \beta = \Gamma_N . \tag{A.61}$$

In the inextensional theory, the displacement gradient Γ_T and its derivative are eliminated terms of gradient Γ_N and its derivative as shown by eqs. (59) to (61). We assume for an extensional arch theory with strains small with respect to unity that the rotation β and its gradient are given with sufficient accuracy from eqs. (59) to (61). Equation (60) implies the range of rotations considered in the theory is $-\pi/2 < \beta < +\pi/2$. If $|\beta| = \pi/2$, then $\Gamma_N = 1$, $\Gamma_T = -1$, and $\Gamma'_N = 0$ so that eq. (59) for β' becomes an indeterminate form. Consideration of eq. (54), and that the strains are assumed to be small compared to unity, implies $|h\beta'|$ should also be small compared to unity. Finally, since $\kappa_o^* - \kappa_o = \beta'$ then $\delta\kappa_o^* = \delta\beta'$ and the internal virtual work (55) becomes

$$\delta W_{int} = \int_0^S (N\delta\varepsilon_o + M\delta\beta')\, ds_o . \tag{A.62}$$

Hooke's Law

For a linear elastic, isotropic material, and neglecting the transverse normal stresses with respect to stress S_{ss} for the arch, Hooke's law is simply

$$S_{ss} = E\ \varepsilon \tag{A.63}$$

in which E is the modulus of elasticity. Substituting eq. (54) for the strain in (61), and substituting this result into eqs. (56), we get

$$\begin{Bmatrix} N \\ M \end{Bmatrix} = \begin{bmatrix} E\bar{A} & -E\bar{I}\kappa_o \\ -E\bar{I}\kappa_o & E\bar{I} \end{bmatrix} \begin{Bmatrix} \varepsilon_o \\ \beta' \end{Bmatrix} \tag{A.64}$$

where we have taken the reference arc to pass through the centroid of the cross section such that

$$\int_A \zeta dA = 0 . \tag{A.65}$$

Quantities $\bar{A}$ and $\bar{I}$ in eqs. (64) are defined by

$$\bar{A} = \int_A \frac{dA}{1 - \zeta\kappa_o} , \qquad \bar{I} = \int_A \frac{\zeta^2 dA}{1 - \zeta\kappa_o} . \tag{A.66}$$

Summary for the Deep Arch

Equilibrium of the arch in the deformed state is imposed by the principle of virtual work. From eqs. (2), (33), (36), and (37), the external virtual work is

$$\delta W_{ext} = P\delta w_m + K(\ell - \ell_o)\left[\frac{\ell_o - w_m}{\ell} \delta w_m - \frac{u_m}{\ell} \delta u_m \right] \tag{A.67}$$

in which the subscript m means midspan. The length of the spring in the deformed configuration is determined from the midspan displacements by the positive square root of eq. (35). Thus,

$$\ell = \sqrt{(\ell_o - w_m)^2 + u_m^2} . \tag{A.68}$$

The internal virtual work referenced to the initial state is given by eq. (62) which is repeated below.

$$\delta W_{int} = \int_0^S (N\delta\varepsilon_o + M\delta\beta')\, ds_o . \tag{A.62}$$

Equation (11) for the strain of the reference arc, eq. (59) for the rotation gradient, and eqs. (6) for the components of the gradient of the displacement vector are repeated below for convenience.

$$\varepsilon_o = \Gamma_T + \frac{1}{2}\Gamma_T^2 + \frac{1}{2}\Gamma_N^2 \tag{A.11}$$

$$\beta' = \Gamma'_N / \sqrt{1 - \Gamma_N^2} \tag{A.59}$$

$$\left.\begin{aligned} \Gamma_T &= u'_o - \kappa_o w_o \\ \Gamma_N &= w'_o + \kappa_o u_o \end{aligned}\right\} . \tag{A.6}$$

The hoop force N and bending moment M are related to strain ε_o and rotation gradient β' by eq. (64).

SHALLOW CIRCULAR ARCH

Let R_o $(= 1/\kappa_o)$ denote the radius of the reference arc, α the semi-opening angle of the reference arc, L the span between supports, and let H denote the midspan rise of the arch above the line connecting the supports. See Fig. 1. A shallow arch is defined by a small rise to span ratio H/L. For the circular arch

$$H/L = \alpha/4 + 0(\alpha^3) \tag{A.69}$$

so that a shallow arch is characterized by a small semi-opening angle. The maximum slope of the reference arc with respect to the line passing through the supports is α and occurs at the support points. Since α is small, the slope of the reference arc is small everywhere on the arc. We move the origin of the x-axis to a position midway

between the supports with respect to the deep arch development. Thus $-L/2 \leq x \leq L/2$. The symmetric function $z_o(x)$ describes the reference arc in the initial state. A shallow circular arch is described by the quadratic function.

$$z_o(x) = H\left[1 - (2x/L)^2\right], \qquad x \in (-L/2, L/2). \tag{A.70}$$

The initial configuration is described by the position vector

$$\underset{\sim}{r}_o(x) = x\,\hat{\underset{\sim}{i}} + z_o(x)\,\hat{\underset{\sim}{k}}. \tag{A.71}$$

We prefer to use cartesian coordinates for the shallow arch rather than the curvilinear coordinates (s_o, ζ) used in the deep arch development, as is suggested by eq. (71). From eq. (71), the differential arc length ds_o, unit tangent and normal vectors to the reference arc, and the curvature are

$$ds_o = \sqrt{1 + (z'_o)^2}\ dx \tag{A.72}$$

$$\hat{\underset{\sim}{T}} = \left[\hat{\underset{\sim}{i}} + z'_o\,\hat{\underset{\sim}{k}}\right]/\sqrt{1 + (z'_o)^2} \tag{A.73}$$

$$\hat{\underset{\sim}{N}} = \left[z'_o\,\hat{\underset{\sim}{i}} - \hat{\underset{\sim}{k}}\right]/\sqrt{1 + (z'_o)^2} \tag{A.74}$$

$$\kappa_o = -z''_o/\left[1 + (z'_o)^2\right]^{3/2} \tag{A.75}$$

in which a prime now means derivative with respect to x. For a shallow arch

$$0 < |z'_o| \leq \alpha << 1. \tag{A.76}$$

Thus eqs. (72) to (75) simplify to

$$ds_o = dx \tag{A.77}$$

$$\hat{T} = \hat{\underset{\sim}{i}} + z'_o\,\hat{\underset{\sim}{k}} \tag{A.78}$$

$$\hat{N} = z'_o \, \hat{\underset{\sim}{i}} - \hat{\underset{\sim}{k}} \tag{A.79}$$

$$\kappa_o = -z''_o \, . \tag{A.80}$$

The deformed configuration of the reference arc is given by the position vector

$$\underset{\sim}{r}^{\cdot}_o = \underset{\sim}{r}_o + \underset{\sim}{\Delta}_o \tag{A.81}$$

In which the cartesian representation of the displacement vector is

$$\underset{\sim}{\Delta}_o = u(x) \, \hat{\underset{\sim}{i}} + w(x) \, \hat{\underset{\sim}{k}} \, . \tag{A.82}$$

In eq. (82), u and w are the x- and z-direction displacements, respectively, of a point on the reference arc. Note that w(x) is positive vertically upward toward the convex side of the reference arc, whereas $w_o(x)$ for the deep arch was defined positive toward the concave side of the reference arc. The line element $d\underset{\sim}{r}_o$ in the initial configuration displaces, rotates, and stretches to line element $d\underset{\sim}{r}^{\cdot}_o$ in the deformed configuration. The differential of $\underset{\sim}{r}^{\cdot}_o$ in eq. (81), using eqs. (71) and (82), is

$$d\underset{\sim}{r}^{\cdot}_o = \left[(1 + u') \, \hat{\underset{\sim}{i}} + (z'_o + w') \, \hat{\underset{\sim}{k}} \right] dx \, . \tag{A.83}$$

The square of the magnitude of this differential vector is

$$\left(ds^{\cdot}_o\right)^2 = \left[(1 + u')^2 + (z'_o + w')^2 \right] dx^2 \, . \tag{A.84}$$

The stretch ratio λ_o defined in eq. (8) is determined by eqs. (72) and (84) to be

$$\lambda_o^2 = \frac{(1 + u')^2 + (z'_o + w')^2}{1 + (z'_o)^2} \tag{A.85}$$

and the Green's strain defined in eq. (9) is

$$\varepsilon_o = \frac{u' + z'_o w' + 1/2(u')^2 + 1/2(w')^2}{1 + (z'_o)^2} . \tag{A.86}$$

From eqs. (83), (84), and (72) the unit tangent vector to the reference arc in the deformed state, defined by the first of eqs. (12), is

$$\hat{\underset{\sim}{t}} = \frac{(1 + u')\hat{\underset{\sim}{i}} + (z'_o + w')\hat{\underset{\sim}{k}}}{\lambda_o\sqrt{1 + (z'_o)^2}} . \tag{A.87}$$

Since the unit normal to the reference arc in the deformed state is rotated 90° clockwise with respect to $\hat{t}$, we get from eq. (87) that

$$\hat{\underset{\sim}{n}} = \frac{(z'_o + w')\hat{\underset{\sim}{i}} - (1 + u')\hat{\underset{\sim}{k}}}{\lambda_o\sqrt{1 + (z'_o)^2}} . \tag{A.88}$$

The rotation of line element $d\underset{\sim}{r}_o$ to $d\underset{\sim}{r}^*_o$ was defined by the angle β in eq. (14). Trigonometric functions of angle β are determined from eqs. (73), (74), and (87). The result is

$$\lambda_o \cos\beta = 1 + \Gamma_T \tag{A.89}$$

$$\lambda_o \sin\beta = \Gamma_N \tag{A.90}$$

in which we define

$$\Gamma_T = [u' + z'_o w'][1 + (z'_o)^2]^{-1} \tag{A.91}$$

$$\Gamma_N = [-w' + z'_o u'][1 + (z'_o)^2]^{-1} . \tag{A.92}$$

The quantities Γ_T and Γ_N above are the tangential and normal components of the displacement gradient vector, $d\underset{\sim}{\Delta}_o/ds_o$, in terms of the shallow arch variables. The

curvature κ_o^* on the reference arc in the deformed state is defined in terms of the rotation β by eq. (17). Using the chain rule and eq. (72), the eq. (17) is rewritten as

$$\lambda_o \kappa_o^* = \beta' \left[1 + (z'_o)^2 \right]^{-1/2} + \kappa_o \tag{A.93}$$

where again the prime now means derivative with respect to x. The rotation gradient β' is determined by differentiating eqs. (89) and (90) with respect to x and solving the resulting two simultaneous equations for $\lambda_o \beta'$ and λ'_o. This procedure is analogous to that followed for the deep arch (see eqs. (15), (16), and (58)). The result is

$$\begin{aligned} \lambda_o \beta' &= (\cos \beta) \Gamma'_N - (\sin \beta) \Gamma'_T \\ \lambda'_o &= (\sin \beta) \Gamma'_N + (\cos \beta) \Gamma'_T . \end{aligned} \tag{A.94}$$

For strains small with respect to unity, i.e., $0 < |\varepsilon_o| << 1$, the rotation gradient β' may be approximated in an analogous fashion to the deep arch (see the discussion of eq. (59)). From eqs. (94) under the assumption of small strains, we approximate β' as

$$\beta' = \frac{\Gamma'_N}{\cos \beta} . \tag{A.95}$$

For shallow arches eq. (76) is applicable for the initial configuration, but we also assume that the rotation β is the same order of magnitude as the semi-opening angle α. That is, the inverted configuration of the arch, or its approximate snapped-through configuration, would have an end rotation of approximately 2α. Thus, for α small and respect to unity, the rotation β is small with respect to unity, and we can use the small angle approximation for the trigonometric functions of β such that $\cos \beta \simeq 1$ and $\sin \beta \simeq \beta$. In addition, strains small with respect to unity imply $\lambda_o \simeq 1$ in eqs. (89) and (90). Consequently, for small strains and rotations the order of the semi-opening angle we approximate the rotation angle by

$$\beta \simeq \Gamma_N . \tag{A.96}$$

The approximation for the rotation gradient from eq. (95) becomes

$$\beta' \simeq \Gamma'_N \tag{A.97}$$

which is consistent with eq. (96). Since the approximations imply $1 + \overline{\Gamma}_T \sim 1$ in (89), and that the strain ε_o in (86) is equivalent to

$$\varepsilon_o = \Gamma_T(1 + 1/2\ \Gamma_T) + 1/2\ \Gamma_N^2 , \tag{A.98}$$

it is consistent to approximate the strain-displacement relation (98) by

$$\varepsilon_o \simeq \Gamma_T + 1/2\ (\Gamma_N)^2 . \tag{A.99}$$

Finally, for shallow arches we neglect the contribution of the displacement u with respect to w in rotation in eq. (96). This implies from eq. (92) that

$$\Gamma_N \simeq -w' \tag{A.100}$$

in which we also neglected the initial slope angle with respect to unity. In terms of displacements, the approximations to the strains measures for a shallow arch undergoing small strains are

$$\varepsilon_o = u' + z'_o w' + 1/2(w')^2 \tag{A.101}$$

$$\beta = -w' \tag{A.102}$$

$$\beta' = -w'' \tag{A.103}$$

$$\kappa_o^{\cdot} - \kappa_o = \beta' . \tag{A.104}$$

The internal virtual work for the shallow arch is the same expression (62) developed for the deep arch except that the strains are given by eqs. (101) and (103) and

$ds_o \simeq dx$ (see eq. (77)), where $-L/2 \leq x \leq L/2$. We assume Hooke's law for the shallow arch is the same as for a straight beam, such that

$$N = EA\ \varepsilon_o\,, \qquad M = EI\ \beta' \tag{A.105}$$

where A is the cross-sectional area and I is the second area moment. The expressions for A and I are given in eqs. (66) for $\bar{A}$ and $\bar{I}$, respectively, except $1 - \zeta\kappa_o$ is replaced by unity in the denominator of the integrands. Since the arch is linearly elastic the internal virtual work is the negative of first variation of the strain energy U; i.e.,

$$\delta W_{int} = -\delta U \tag{A.106}$$

where

$$U = \frac{1}{2}\int_{\frac{-L}{2}}^{\frac{+L}{2}} (N\varepsilon_o + M\beta')\ dx\,. \tag{A.107}$$

The external loads P and F_s acting at midspan are also conservative mechanical forces. The applied load P is assumed to be a deadweight load. Thus, the potential function Ω for the external loads is

$$\Omega = Pw(0) + 1/2K(\ell - \ell_o)^2\,. \tag{A.108}$$

The axial displacement u is smaller than the lateral displacement w for the shallow arch, so that we neglect the contribution of u in computing the elongation of the spring; see eqs. (67) and (68). Since w at midspan for the shallow arch is positive in the opposite sense with respect to the deep arch development, we approximate eq. (68) as

$$\ell \simeq \ell_o + w(0) \tag{A.109}$$

such that $\ell - \ell \simeq w(0)$ in the external potential (108). The total potential energy V for the shallow arch and load configuration is the sum of the strain energy (107) and the external potential (108). That is,

$$V = \frac{1}{2}\int_{\frac{-L}{2}}^{\frac{L}{2}} (N\ \varepsilon_o + M\beta')\ dx + P\ w(0) + \frac{1}{2} K w^2(0) \qquad (A.110)$$

in which the strain ε_o is given by eq. (101), rotation gradient β' by (103), and resultants N and M by eqs. (105).

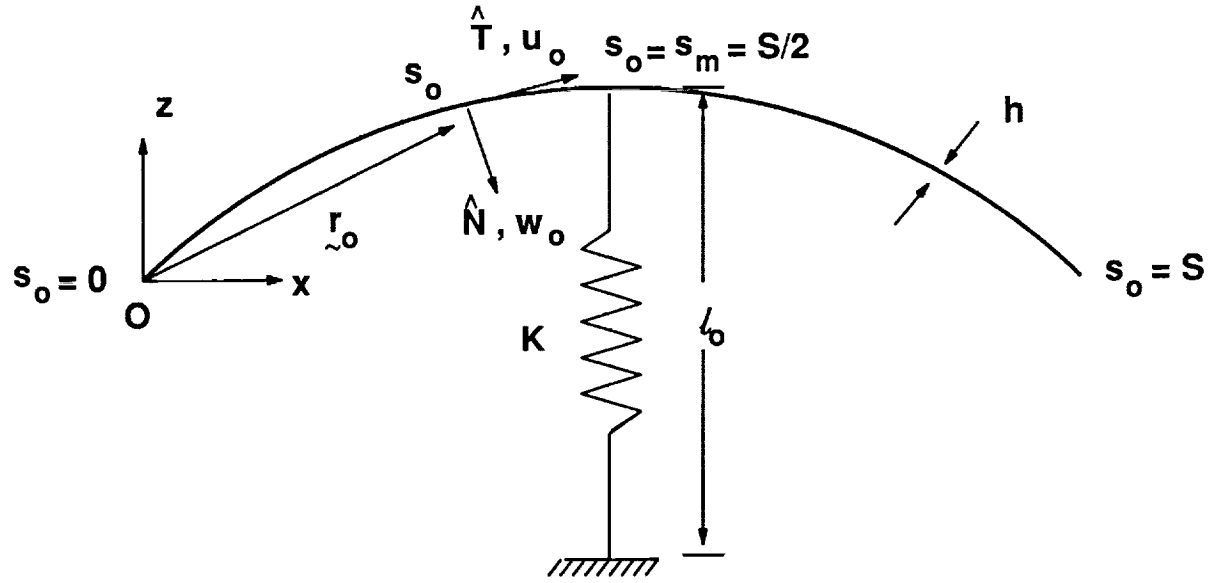

Fig. A.1 Initial configuration of the arch.

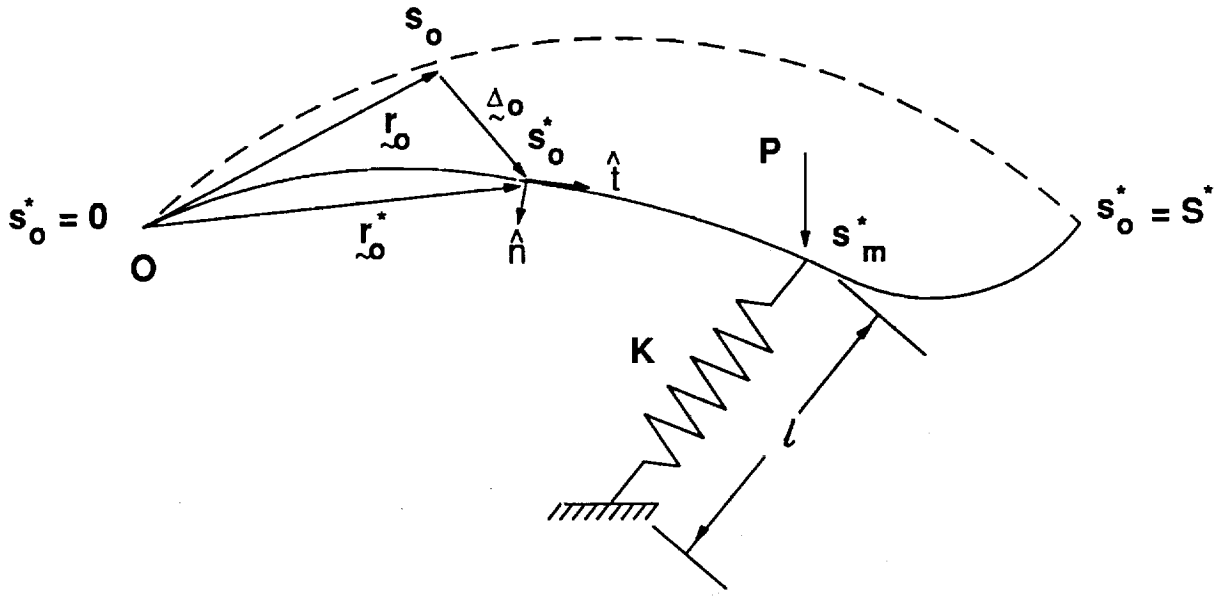

Fig. A.2 Deformed configuration of the arch.

Appendix B

User's Guide for FENLA FORTRAN

FENLA is a program which implements the finite-element formulation of the solution to the problem of a simply-supported general arch (either shallow or deep) with a spring attached at centerspan subjected to a centerspan point load. The solution is developed from the principle of virtual work as outlined in the text, eqs. 61-97, and uses Newton's method to solve the system of non-linear equations. The general response of such an arch-spring system, loaded from an undeformed shape at zero load, is for the arch midspan to deflect downward with the normal deflections, w, of the rest of the arch being symmetric with respect to the midspan. As the load increases the midspan deflection also increases. A curve tracing the load-midspan deflection for this initial response is known as the symmetric equilibrium path. For some arches, depending on the physical properties of the arch-spring system, a point, called a limit point, will be reached, at which for the centerspan deflection to continue to increase smoothly, without a large jump in the deflection, the applied load must be decreased. The symmetric equilibrium path can be traced past a limit point with the load decreasing with increasing midspan deflection. For some arches another limit point on the symmetric equilibrium path will be reached, after which the load will again increase with increasing midspan deflection. Although in general several symmetric equilibrium paths may exist for an arch-spring system they are not necessarily joined by limit points. Some symmetric equilibrium configurations are disjoined, meaning that one configuration cannot be reached from another by a smooth stepwise deformation of the arch, but can only be obtained by a significant change in arch shape. Another possible arch response, again depending on the physical properties of the system, corresponds to an equilibrium path that bifurcates off the symmetric equilibrium path. Along this path the arch responds with continued increase in midspan deflection, the load either increasing or decreasing depending on the physical properties of the arch-spring system. However, the w deflections of

the arch are asymmetric about the midspan. If allowed to follow this path far enough the arch may begin to exhibit symmetric behavior, indicating that the bifurcated path has intersected a symmetric equilibrium path.

The program is intended to be used to trace the symmetric equilibrium path, from zero load and an undeformed shape, indicating the possible existence of critical or limit and bifurcation points. The program can then be used to find a solution on the adjacent equilibrium path, either past the limit point, or on the bifurcated path. Once a single solution is known on the adjacent equilibrium path that path can also be traced.

The program is started with an initial applied load and an initial guess at the displacement vector, relative to the undeformed arch shape, for that load. An incremental displacement vector is solved for and added to the initial guess until the solution converges to the displacement vector for the applied load. Convergence occurs when the internal virtual work of the arch-spring system equals the external virtual work for the applied load. If the program is run in batch mode, once the solution at the current load is found the load is incremented and the displacement vector just found is used as the initial guess of the displacement vector at the new load. This procedure continues until either the load limit specified by the user is reached or until the program cannot converge to a solution at the new load within ten iterations. In batch mode the program indicates the existence of critical points, points where an eigenvalue of the tangent stiffness matrix becomes zero. Critical points are points such as limit points and bifurcation points where another equilibrium configuration exists in addition to the current equilibrium configuration. Bifurcation points are indicated by an eigenvalue passing through zero, while limit points are indicated by the inability of the program to find a solution at the next load step, with an eigenvalue that is approaching zero as the load approaches the limit point load.

The interactive mode of the program is particularly useful at critical points, where it can be used to find a solution on an adjacent equilibrium path, i.e., past a limit point, or on the path continuing from a bifurcation point. To find a solution on the adjacent path at the critical point the arch shape must be perturbed. This perturbation is accomplished by using the eigenvector associated with the eigenvalue that is zero at the critical point. Because at a limit point the adjacent path is a continuation of the current equilibrium path the eigenvector will have the same shape as the current equilibrium configuration, i.e., it will be symmetric. At a bifurcation point the eigenvector associated with the zero eigenvalue will have a different shape than the current equilibrium configuration, i.e., it will be asymmetric. At either type of critical point the critical eigenvector can be scaled and added to the current displacement vector to be used as a guess at the displacement vector on the adjacent path.

The Program

Main Program: The program consists of a main program with several subroutines. In the main program the problem is set up and the system of linear incremental equations is solved using an IMSL subroutine. The error in the solution is checked and based on the error the program either iterates to find a new solution or reports the new solution. In batch mode the program checks the eigenvalues of the tangent stiffness matrix at each solution to indicate the possible existence of critical points. A bifurcation point is indicated by an eigenvalue, with an associated asymmetric eigenvector, that passes through zero. The program estimates the bifurcation load by using a linear interpolation to find the load at which the eigenvalue is actually zero. The displacement vector at a load very near the bifurcation point is then found. A possible limit point is indicated when the program cannot find a solution for the next load step even after reducing the load increment by half three times. As a limit point is approached one of the eigenvalues of the tangent stiffness matrix will approach zero. The eigenvector associated with this eigenvalue will be symmetric. The sym-

metry or asymmetry of the eigenvector is determined by looking at the elements of the eigenvector corresponding to u and w' of the midspan of the arch. If these elements are zero the eigenvector is said to be symmetric, if either of them is nonzero the eigenvector is said to be asymmetric.

In interactive mode once a solution is determined the program will provide the user with information about the eigenvectors of the tangent stiffness matrix and will ask the user to select an eigenvector and a scale factor by which to multiply the eigenvector when adding it to the previous displacement vector to use as a guess at the displacement vector on the adjacent equilibrium path. The interactive mode can also be used to continue along the current equilibrium path by specifying the value of the scale factor to be zero, thus using the current displacement vector as the initial guess at the displacement vector at a different load.

The Subroutines: The subroutines consist of several subroutines to compute the elements of the stiffness matrix of the problem, several subroutines for general matrix manipulation, and a subroutine to add the effect of a centerspan spring to the problem. The stiffness matrix is computed by numerical integration, using Simpson's rule, of the matrices derived in the text, eqs. 93, 94, 95. The integrands, which are functions of the shape function for the element used, see fig. 9 and eqs. 77-81, were computed using MACSYMA and then used in this program. Several of the subroutines are the MACSYMA FORTRAN77 output.

Program Input

The program has two input files. The file corresponding to unit 11 will contain physical information about the arch, the load limits for the run, and information about the particular run, i.e., whether the run is interactive or batch, and whether or not the initial guess at the displacement vector will be input from a file. The file corresponding

to unit 14 will contain a value of load and the initial guess of the global displacement vector (GU). If the initial guess at the displacement vector will be the zero vector unit 14 need not be used. Unit 5 is used for input from the terminal during an interactive run. The user will be prompted for necessary information. The number of elements in the model is set using the parameter statement at the beginning of the program. The smallest number of elements possible is 4. The number of elements is increased by cutting each element in half so that possible models can have 4, 8, 16, 32, ... elements. The number of eigenvalues and eigenvectors of the tangent stiffness matrix to be computed can also be changed in the parameter statement. The input from each of units 11, 14, and 5 is outlined below.

Unit 11 (Variable names are given parentheses)

- Flag (INTERACT) for batch (0) or interactive (1) mode. (line 1)
- The physical parameters E*A (EA) in pounds, E*I (EI) in pound*in^2, and curvature (C) in inches^{-1} for the arch. The arch is assumed to be circular with constant curvature. (line 2)
- The arch length (SL) and span length (RL) of the arch in inches. (line 3)
- The spring stiffness (SK) in pounds/inch, and original spring length (SPRL) in inches. (line 4)
- The distributed load (Q) in pounds/inch. The program can handle a distributed load in addition to the point load if desired. (line 5)
- The initial point load (PINIT), the final load (PFIN), and the load increment (DP) in pounds. Note that this information is used only in batch mode. (line 6)
- The number of Simpson's integration intervals (NSI) to use in the numerical integration of the stiffness matrix. (line 7)
- A scale factor (SFAC) to use in the solution of the system of equations as necessary to prevent numerical (underflow and overflow) problems. (line 8)
- Flag (IREAD) for indicating whether or not to read the initial guess of the displacement vector from an input file (0 if no, 1 if yes). (line 9)

All input is unformatted.

Unit 14

- The load (P) at or near the displacement vector that follows. (line 1)
- The value of the global displacement variable at each node (GU(I)) to be used as the initial guess. (lines 2 to number of global displacement variables (NN) + 1)

Again the input is unformatted. Note that on the element level the nodes are numbered as indicated in fig. 9, with local degrees of freedom GUEL(I,1) = u of element I node 1, GUEL(I,2) = w of element I node 1, GUEL(I,3) = w' of element I node 1, GUEL(I,4) = u of element I node 2, GUEL(I,5) = w of element I node 2, GUEL(I,6) = w' of element I node 2, and GUEL (I,7) = u of element I node 3. On the global level the nodes are numbered from left to right with the global displacement variables being GU(1) = u of node 1, GU(2) = w of node 1, GU(3) = w' of node 1, GU(4) = u of node 2, GU(5) = u of node 3, GU(6) = w of node 3, GU(7) = w' of node 3, and so on.

Unit 5

In interactive mode the user will be asked to respond to several questions. The user will be asked if he wants to update PLAST, EVLAST, GULAST, and PHELAST. These variables are used as the base from which to try to find a solution on the adjacent equilibrium path. They correspond to the load, eigenvalues, displacement vector, and eigenvectors at a solution. The user should answer yes (1) if he wants to use the current solution as the point from which to try to move to a bifurcated path or past a limit point. The user should also answer yes just before quitting the interactive session as the values saved in these variables are written to an output file 13 when the program stops. The user must also respond whether he wants to stop the program (0) or try to find a new solution (1). If he wants to find a new solution he must enter a value of the load, P, the scale factor (EPS) by which to scale the normalized eigenvector, and which eigenvector he wants to use to increment the displacement

vector to obtain an initial guess at the solution on the adjacent path. Note that symmetric eigenvectors are normalized by the term corresponding to w at the center node while asymmetric eigenvectors are normalized by the term corresponding to w at the quarter points of the arch.

Program Output

The program has several output files.

Unit 6 is used for error output in the batch mode and for output to the screen in interactive mode.

Unit 8 contains the nondimensionalized load and midspan deflection for each solution. The nondimensionalization can either be based on the nondimensionalization used in studying the shallow arch problem analytically as outlined in eq. 16 or using nondimensionalization of Huddleston given in eq. 100.

Unit 9 contains the general output for the program. The top of the file reports the physical parameters for the run, i.e., the geometrical data for the arch, the initial load, load increment, and final load, information about the spring, and information about the number of nodes and elements in the model. The program then outputs to this file the error in each type of displacement variable (u, w, and, w') for each iteration as a solution is searched for at each load step. When a solution has converged the program prints the number of iterations required, the load level, the solution displacement vector, and the lowest three eigenvalues of the tangent stiffness matrix at the solution. In batch mode when the lowest eigenvalue becomes negative, as it will if the current load and solution are on a part of the equilibrium path that is either past a bifurcation point or past a limit point, the program outputs the eigenvectors associated with the three lowest eigenvalues. In batch mode unit 9 also contains information on the loads at which critical points are suspected. In interactive mode the

program outputs the smallest three eigenvalues and the associated eigenvectors for each solution found.

Unit 13 is a file containing the load and displacement vector for a couple of key load steps. In batch mode it will contain the load and displacement for the load closest to but just before a suspected bifurcation point, and the load and displacement vector for the last load level before a suspected limit point. In interactive mode unit 13 will contain the load and displacement vector stored as PLAST and GULAST when the user ends the program. This file is set up to be used as the input file 14 on subsequent run.

Unit 15 is a file containing the load and the lowest eigenvalue at each solution. It is used for plotting a load vs. eigenvalue curve to help indicate bifurcation points.

Sample Problems

Example Batch Run

This example is for a shallow aluminum arch, with a 1 inch wide by 0.1 inch thick cross section. The arch span is 3.97 inches and the arch rise is 0.0866 inches. The stiffness of the centerspan spring is 213 pounds/in. This batch run will start at zero load and zero displacement and find the displacement solution vector for increasing load levels.

Input file 11

Line Number	Variables	Value
Line 1:	INTERACT	0
Line 2:	EA,EI,C	1.D6 833.33 0.04391
Line 3:	SL,RL	3.975 3.97
Line 4:	SK,SPRL	213.0 0.3

Line 5:	Q	0.0
Line 6:	PINIT,PFIN,DP	20.0 200.0 5.0
Line 7:	NSI	8
Line 8:	SFAC	1.D-5
Line 9:	IREAD	0

When IREAD = 0 the additional data file (unit 14) containing an initial guess is not needed.

Output

The output (unit 9) from this run would consist of the load and displacement vector at 5 pound increments from 20 lbs. to 90 lbs. At 95 lbs. the program is not able to find a solution using the displacement vector at 90 lbs. as an initial guess. The program will reduce the load increment (DP) by half and try to find a solution for P = 92.5 again using the displacement vector at 90 lbs. as the initial guess at the displacement vector. At 92.5 lbs. the program is again unable to find a solution so the load increment is again reduced by half. The program will reduce DP by half three times, after which the batch run will stop, reporting a suspected limit point. The user will note that the lowest eigenvalue of the tangent stiffness matrix at each solution is approaching zero as the load approaches the suspected limit point. For the last load before the suspected limit point at which the program can find a solution, the eigenvector associated with the lowest eigenvalue is symmetric. Recall that this indicates that the adjacent path at the critical point has the same shape as the current solution.

Sample Interactive Run

This is an example of the input files that would be used to try to move past the limit point for the arch in the batch example above. The number of lines in input file 14

depends on the number of elements and nodes in the model. The data file below corresponds to a run with 16 elements, 33 nodes, and 67 degrees of freedom. The parameter statement in the FORTRAN file that corresponds to this case would be uncommented with all other parameter statements commented out.

Input file 11

Line Number	Variables	Value
Line 1:	INTERACT	1
Line 2:	EA,EI,C	1.D6 833.33 0.04391
Line 3:	SL,RL	3.975 3.97
Line 4:	SK,SPRL	213.0 0.3
Line 5:	Q	0.0
Line 6:	PINIT,PFIN,DP	20.0 200.0 5.0
Line 7:	NSI	8
Line 8:	SFAC	1.D-5
Line 9:	IREAD	1

Input file 14

Line Number	Variables	Value
Line 1:	P	0.9125000E + 02
Line 2:	GU(1), u(1)	0.0000000E + 00
Line 3:	GU(2), w(1)	0.0000000E + 00
Line 4:	GU(3), w'(1)	-0.3398966E-01
Line 5:	GU(4), u(2)	-0.1825044E-03
Line 6:	GU(5), u(3)	-0.3425904E-03
Line 7:	GU(6), w(3)	0.8474547E-02
Line 8:	GU(7), w'(3)	-0.3431770E-01
Line 9:	GU(8), u(4)	-0.4805153E-03
Line 10:	GU(9), u(5)	-0.5959346E-03

Line 11:	GU(10), w(5)	0.1705960E-01
Line 12:	GU(11), w'(5)	-0.3474200E-01
Line 13:	GU(12), u(6)	-0.6882522E-03
Line 14:	GU(13), u(7)	-0.7564230E-03
Line 15:	GU(14), w(7)	0.2567705E-01
Line 16:	GU(15), w'(7)	-0.3444491E-01
Line 17:	GU(16), u(8)	-0.7993174E-03
Line 18:	GU(17), u(9)	-0.8156363E-02
Line 19:	GU(18), w(9)	0.3405088E-01
Line 20:	GU(19), w'(9)	-0.3266051E-01
Line 21:	GU(20), u(10)	-0.8042857E-03
Line 22:	GU(21), u(11)	-0.7644650E-03
Line 23:	GU(22), w(11)	0.4172702E-01
Line 24:	GU(23), w'(11)	-0.2872985E-01
Line 25:	GU(24), u(12)	-0.6960134E-03
Line 26:	GU(25), u(13)	-0.5996932E-03
Line 27:	GU(26), w(13)	0.4810602E-01
Line 28:	GU(27), w'(13)	-0.2214872E-01
Line 29:	GU(28), u(14)	-0.4774480E-03
Line 30:	GU(29), u(15)	-0.3328459E-03
Line 31:	GU(30), w(15)	0.5248630E-01
Line 32:	GU(31), w'(15)	-0.1260452E-01
Line 33:	GU(32), u(16)	-0.1711844E-03
Line 34:	GU(33), u(17)	-0.3067471E-21
Line 35:	GU(34), w(17)	0.5411497E-01
Line 36:	GU(35), w'(17)	-0.6270848E-18
Line 37:	GU(36), u(18)	0.1711844E-03
Line 38:	GU(37), u(19)	0.3328459E-03

Line 39:	GU(38), w(19)	0.5248630E-01
Line 40:	GU(39), w'(19)	0.1260452E-01
Line 41:	GU(40), u(20)	0.4774480E-03
Line 42:	GU(41), u(21)	0.5996932E-03
Line 43:	GU(42), w(21)	0.4810602E-01
Line 44:	GU(43), w'(21)	0.2214872E-01
Line 45:	GU(44), u(22)	0.6960134E-03
Line 46:	GU(45), u(23)	0.7644650E-03
Line 47:	GU(46), w(23)	0.4172702E-01
Line 48:	GU(47), w'(23)	0.2872985E-01
Line 49:	GU(48), u(24)	0.8042857E-03
Line 50:	GU(49), u(25)	0.8156363E-03
Line 51:	GU(50), w(25)	0.3405088E-01
Line 52:	GU(51), w'(25)	0.3266051E-01
Line 53:	GU(52), u(26)	0.7993174E-03
Line 54:	GU(53), u(27)	0.7564230E-03
Line 55:	GU(54), w(27)	0.2567705E-01
Line 56:	GU(55), w'(27)	0.3444491E-01
Line 57:	GU(56), u(28)	0.6882522E-03
Line 58:	GU(57), u(29)	0.5959346E-03
Line 59:	GU(58), w(29)	0.1705960E-01
Line 60:	GU(59), w'(29)	0.3474200E-01
Line 61:	GU(60), u(30)	0.4805153E-03
Line 62:	GU(61), u(31)	0.3425904E-03
Line 63:	GU(62), w(31)	0.8474547E-02
Line 64:	GU(63), w'(31)	0.3431770E-01
Line 65:	GU(64), u(32)	0.1825044E-03
Line 66:	GU(65), u(33)	0.0000000E+00

Line 67:	GU(66), w(33)	0.0000000E + 00
Line 68:	GU(67), w'(33)	0.3398966E-01

In the above variable listing for unit 14 the second variable indicates the physical degree of freedom corresponding to each global degree of freedom.

Input To and Output From the Terminal

Input from and output to the terminal in this interactive run could be as follows:

Program:
```
 SOLUTION FOUND FOR P =  0.9125000E + 02
 CURRENT SOLN HAS P =  0.9125000E + 02  W(0) =  0.5411497E-01
 EIGENVECTOR 1 IS SYMMETRIC
 EIGENVECTOR 2 IS ASYMMETRIC
 LAST SOLN HAS P =  0.0000000E + 00  W(0) =  0.0000000E + 00
 VALUE OF EPS =  0.0000000E + 00
 DO YOU WANT TO UPDATE PLAST, EVLAST, GULAST, AND PHELAST? (0 = NO, 1 = YES)
 (IF FIRST TIME THROUGH SAY YES)
```

User:
```
        1
```

Program:
```
 DO YOU WANT TO GUESS AT P AND EPS? OR STOP?
 (0 = STOP, 1 = GUESS)
```

User:
```
        1
```

Program:
```
 ENTER P AND EPS AND WHICH EIGENVECTOR TO USE
```

User:
```
  91.5     0.0         1
```

Program:
```
 SOLUTION FOUND FOR P =  0.9150000E + 02
 CURRENT SOLN HAS P =  0.9150000E + 02  W(0) =  0.5565260E-01
 EIGENVECTOR 1 IS SYMMETRIC
 EIGENVECTOR 2 IS ASYMMETRIC
 LAST SOLN HAS P =  0.9125000E + 02  W(0) =  0.5411497E-01
 VALUE OF EPS =  0.0000000E + 00
 DO YOU WANT TO UPDATE PLAST, EVLAST, GULAST, AND PHELAST? (0 = NO, 1 = YES)
 (IF FIRST TIME THROUGH SAY YES)
```

User:
```
        1
```

Program:
```
 DO YOU WANT TO GUESS AT P AND EPS? OR STOP?
```

(0 = STOP, 1 = GUESS)

User:
1

Program:
ENTER P AND EPS AND WHICH EIGENVECTOR TO USE

User:
91.5 0.001 1

Program:
SOLUTION FOUND FOR P = 0.9150000E + 02
CURRENT SOLN HAS P = 0.9150000E + 02 W(0) = 0.5565260E-01
EIGENVECTOR 1 IS SYMMETRIC
EIGENVECTOR 2 IS ASYMMETRIC
LAST SOLN HAS P = 0.9150000E + 02 W(0) = 0.5565260E-01
VALUE OF EPS = 0.1000000E-02
DO YOU WANT TO UPDATE PLAST, EVLAST, GULAST, AND PHELAST? (0 = NO, 1 = YES)
(IF FIRST TIME THROUGH SAY YES)

User:
0

Program:
DO YOU WANT TO GUESS AT P AND EPS? OR STOP?
(0 = STOP, 1 = GUESS)

User:
1

Program:
ENTER P AND EPS AND WHICH EIGENVECTOR TO USE

User:
91.5 0.003 1

Program:
SOLUTION FOUND FOR P = 0.9150000E + 02
CURRENT SOLN HAS P = 0.9150000E + 02 W(0) = 0.5565260E-01
EIGENVECTOR 1 IS SYMMETRIC
EIGENVECTOR 2 IS ASYMMETRIC
LAST SOLN HAS P = 0.9150000E + 02 W(0) = 0.5565260E-01
VALUE OF EPS = 0.3000000E-02
DO YOU WANT TO UPDATE PLAST, EVLAST, GULAST, AND PHELAST? (0 = NO, 1 = YES)
(IF FIRST TIME THROUGH SAY YES)

User:
0

Program:
DO YOU WANT TO GUESS AT P AND EPS? OR STOP?
(0 = STOP, 1 = GUESS)

User:
1

Program:
ENTER P AND EPS AND WHICH EIGENVECTOR TO USE

User:
91.5 0.006 1

Program:
SOLUTION FOUND FOR P = 0.9150000E+02
CURRENT SOLN HAS P = 0.9150000E+02 W(0) = 0.6259679E-01
EIGENVECTOR 1 IS SYMMETRIC
EIGENVECTOR 2 IS ASYMMETRIC
LAST SOLN HAS P = 0.9150000E+02 W(0) = 0.5565260E-01
VALUE OF EPS = 0.6000000E-02
DO YOU WANT TO UPDATE PLAST, EVLAST, GULAST, AND PHELAST? (0=NO, 1=YES)
(IF FIRST TIME THROUGH SAY YES)

User:
1

Program:
DO YOU WANT TO GUESS AT P AND EPS? OR STOP?
(0=STOP, 1=GUESS)

User:
1

Program:
ENTER P AND EPS AND WHICH EIGENVECTOR TO USE

User:
91.25 0.0 1

Program:
SOLUTION FOUND FOR P = 0.9125000E+02
CURRENT SOLN HAS P = 0.9125000E+02 W(0) = 0.6424372E-01
EIGENVECTOR 1 IS SYMMETRIC
EIGENVECTOR 2 IS ASYMMETRIC
LAST SOLN HAS P = 0.9150000E+02 W(0) = 0.6259679E-01
VALUE OF EPS = 0.0000000E+00
DO YOU WANT TO UPDATE PLAST, EVLAST, GULAST, AND PHELAST? (0=NO, 1=YES)
(IF FIRST TIME THROUGH SAY YES)

User:
0

Program:
DO YOU WANT TO GUESS AT P AND EPS? OR STOP?
(0=STOP, 1=GUESS)

User:
0

In this interactive session the program used the information from unit 14 to find an solution for the initial load of 91.25 lbs. The first thing the user does is to update

PLAST, EVLAST, GULAST, and PHELAST so that the program has a displacement vector to use as the starting point for finding additional solutions. The user then indicates that he wants to find a solution for a new load. In this example the user first tries to get a little closer to the limit point by specifying a higher load, 91.5 lbs. and a value of EPS = 0, which means that the solution at 91.25 lbs. will used as the initial guess at the solution at 91.5 lbs., without being perturbed. The program finds a solution at 91.5 lbs. and the user decides to use this displacement vector as the starting point for moving past the limit point so he updates PLAST, EVLAST, GULAST, and PHELAST. He then tries to find a solution past the limit point by specifying the load to again be 91.5 lbs., but perturbing the displacement vector using the symmetric eigenvector, eigenvector 1, and a scale factor of 0.001. The symmetric solution vector is used because, as described earlier, the solution past the limit point should have the same shape as the current solution. The value of EPS = 0.001 is too small to significantly perturb the displacement vector and the program converges to the solution already known. Thus a larger value of EPS is needed. The user tries EPS = 0.003. Note that there was no need to update PLAST, EVLAST, GULAST, and PHELAST. Again, the value of EPS is not large enough to perturb the solution away from the already known solution. Using EPS = 0.006, the program finds a displacement vector for P = 91.5 lbs. which is different from the already known solution as indicated by the larger midspan deflection. This new solution is expected to be a solution on the path past the limit point so PLAST, EVLAST, GULAST, and PHELAST are updated. The user can stop the program at this point, or continue as in the example above. In the example above the user decided to continue along the symmetric path that is past the limit point by specifying a lower load P = 91.25 lbs. and using the solution at 91.5 lbs. that was just found, without perturbing it, i.e., EPS = 0, as the guess at the solution at 91.25 lbs. The program was able to find a solution for P = 91.25 lbs. Note that the midspan deflection is different than the midspan deflection for the same load at the start of the interactive session. The user does not update PLAST, EVLAST, GULAST,

and PHELAST so when the user stops the program in the next step output file 13 will contain information about the load and displacement vector the last time these variables were updated, namely for P = 91.5 lbs., just past the limit point. If the user wishes to continue following the symmetric path that is past the limit point he could start another batch session using the information saved in unit 13 from this interactive session as the input unit 14 changing the load information in unit 11 appropriately.

Miscellaneous Notes

The above example cases were for an arch that exhibits only limit points and no bifurcation points. Arch-spring systems that exhibit both bifurcation and limit points are handled in a similar fashion to the example. In trying to move onto a bifurcated path or past a limit point, the user must specify a value of load (P), the factor (EPS) by which to scale the normalized eigenvalue before using it to increment the displacement vector, and which eigenvector to use to increment the displacement vector to obtain an initial guess at the solution on the adjacent path. For trying to move past a limit point or onto a descending bifurcated path, the value of the load can be the same as the initial load. For trying to move onto an ascending bifurcation path the load must be higher than the last load before the bifurcation point. The value of EPS needed to move onto the adjacent path depends strongly on the physical size of the arch and the closeness of the initial solution to the bifurcation or limit point. The closer to the bifurcation or limit point and the shallower the arch, the smaller the value of EPS that will be needed. For example, the value of EPS needed to move past the limit point for the arch used in the sample interactive file (see also fig. 12) was 0.006, while the value of EPS needed to move past the limit point in fig. 17 was 0.2. To move past limit points, the symmetric eigenvector will be used. To move onto a bifurcated path, an asymmetric eigenvector should be used.

The user should be aware that it is possible to bypass limit points, i.e., effectively move from point L to point M in fig. 2a in one load step, if the load increment is too

C-2

large. It is also possible to end up on a symmetric path even though the initial guess at the displacement vector corresponds to a solution on a bifurcated path if the two solution configurations are similar. Thus the user may need to try several load increments before being able to follow the path he intends to follow.

The development of the governing equations used in the finite-element formulation assumes that the change in slope of the arch from its original slope is at no point greater than 90°. If the change in slope should ever reach 90° the quantity Γ_N^2 will be come greater than one and the quantity β' becomes undefined. Because Γ_N^2 can also become greater than one if there is no solution for a given load in the vicinity of the initial guess for that load, the program alerts the user to the fact that $\Gamma_N^2 > 1$ and therefore argument under the radical sign in one of several expressions is negative and the program cannot continue on along the current path. In general this message indicates that a new initial guess is needed (i.e., a new value of EPS in interactive mode), but in certain cases it could indicate the change in slope is actually greater than 90°. Note also that the development of the governing equations permits tangential displacement and rotation of the center node of the arch.

Appendix C

FENLA FORTRAN Source Code

This appendix contains the FORTRAN source code for FENLA. The code accesses Version 1.1 IMSL (International Mathematical and Statistical Library, Houston, Texas) Math/Library subroutines DLSLSF, DEVLSF, DEVASF, DEVESF, and IWKIN. This version of the code was compiled using the IBM VS FORTRAN Version 2.4 compiler, a FORTRAN77 and FORTRAN66 compiler.

```
C THIS VERSION IS DESIGNED TO EITHER FOLLOW A SYMMETRIC EQUILIBRIUM PATH
C INDICATING POSSIBLE LIMIT POINTS AND BIFURCATION POINTS IN BATCH MODE,
C OR TO INTERACTIVELY TRY TO JUMP TO AN ADJACENT PATH GIVEN A NEW P AND
C EPSILON TO MULTIPLY THE EIGENVECTOR OF THE STIFFNESS MATRIX TO GET A
C NEW GUESS AT THE DISPLACEMENT VECTOR.
C
C NON-LINEAR INCREMENTAL FINITE ELEMENT SOLUTION FOR AN ARCH WITH A
C MIDSPAN SPRING SUBJECTED TO A TRANSVERSE MIDSPAN POINT LOAD.
C QUADRATIC ELEMENTS ARE USED FOR THE AXIAL DISPLACEMENTS.  STIFFNESS
C MATRICES ARE INTEGRATED NUMERICALLY USING THE COMPOSITE SIMPSONS RULE.
C
C *** INPUT FILE (UNIT 11) CONTAINS VALUES FOR
C
C INTERACT(FLAG FOR INTERACTIVE OR BATCH MODE)
C EA(E*A),EI(E*I),C(CURVATURE),
C SL(ARCH LENGTH),RL(SPAN LENGTH)
C SK(SPRING STIFFNESS),SPRL(ORIGINAL SPRING LENGTH)
C Q(DISTRIBUTED LOAD),
C PINIT(INITIAL CENTERSPAN POINT LOAD),PFIN(FINAL LOAD),DP(LOAD STEP)
C NSI(NUMBER OF SIMPSONS RULE APPLICATIONS IN STFINT),AND
C FAC(REDUCING FACTOR TO DECREASE THE MAGNITUDE OF THE NUMBERS IN THE
C    STIFFNESS MATRIX TO AVOID OVERFLOW ERR).
C IREAD(FLAG TO READ NON-ZERO INITIAL GUESS AT U'S)
C
C *** NOTES ***
C PARAMETER STATEMENT MUST BE CHANGED TO CHANGE NUMBER OF NODES IN
C ARCH
C
C NONDIMENSIONALIZATION TO COMPARE TO EXACT SOLUTION OR HUDDLESTON
C MUST BE SELECTED BY COMMENTING OUT THE APPROPRIATE LINES.
C ***************************************************************
C LIST OF VARIABLES
C ***************************************************************
C  NE:          NUMBER OF ELEMENTS
C  NN:          NUMBER OF GLOBAL DISPLACEMENT VARIABLES
C              3/NODE AT ENDS OF ELEMENT, 1/NODE FOR CENTER NODES
C  NRN:         NUMBER OF DOF (NN-4)
C  GU(NN):      GLOBAL NODAL DISPLACEMENTS
C  GUEL(NE,NRN): ELEMENT DISPLACEMENTS (ITH ELEMENT, JTH NODE)
C  U(NRN):      UNKNOWN GLOBAL DISPLACEMENTS (DOF)
C  EU(NRN):     INCREMENTAL U'S SOLVED FOR EACH ITERATION
C  CSTIF(2,2):  CONSTITUTIVE MATRIX
C  S(NE):       ELEMENT ENDPOINTS
C  F(NN):       NODAL FORCES FROM APPLIED POINT LOADS
C  QQ(NN):      NODAL FORCES FROM APPLIED DISTRIBUTED FORCES
C  FF(NN):      TOTAL APPLIED NODAL FORCES
C  P:           MIDSPAN POINT LOAD AT EACH LOAD STEP
C  PINIT:       THE INITIAL POINT LOAD (INPUT)
C  PFIN:        THE POINT LOAD TO STOP AT (INPUT)
C  DP:          THE LOAD STEP - DELTA P (INPUT)
C  GR(NRN):     STIFFNESS FROM THE CONSTITUTIVE RELATION, USED IN
C              CALCULATING THE RESIDUAL VECTOR FOR ERROR CALCULATION
C  GSTIF(NRN,NRN): GLOBAL STIFFNESS MATRIX FOR EU WITH BC APPLIED
C  R(NRN):      RESIDUAL AT EACH STEP = FF-GR
C  REACT(4):    RESULTANT FORCES AT THE NODE AT WHICH BC ARE APPLIED
C  ERR(3):      ERROR IN EACH OF THE THREE TYPES OF DOF
```

```
C  SFAC:        SCALING FACTOR TO AVOID OVERFLOW/UNDERFLOW WHEN SOLVING
C              SYSTEM OF EQUATIONS
C  NSI:         NUMBER OF SIMPSONS RULE APPLICATIONS IN NUMERICAL INT
C  NPE:         NUMBER OF NODES PER ELEMENT
C  SL:          LENGTH OF ARCH ALONG ARCH
C  RL:          SPAN OF ARCH
C  EA:          PHYSICAL CONSTANT (E*A)
C  EI:          PHYSICAL CONSTANT (E*I)
C  C:           PHYSICAL PARAMETER (CURVATURE OF ARCH)
C  HEL:         LENGTH OF ELEMENT (CONSTANT FOR ALL)
C  SK:          SPRING STIFFNESS
C  SPRL:        ORIGINAL SPRING LENGTH
C  NUMEI:       NUMBER OF EIGENVALUES TO COMPUTE
C  NEVEC:       FLAG INDICATING WHICH EIGENVECTOR TO USE TO ADJUST U
C  EVAL(NUMEI):  EIGENVALUES
C  EVEC(NRN,NUMEI):EIGENVECTORS
C  PHE(NRN,2):   NONDIMENSIONALIZED EIGENVECTOR USED TO ADJUST U
C  PNON:        NONDIMENSIONALIZED LOAD
C  WNON:         NONDIMENSIONALIZED MIDSPAN DEFLECTION
C  NEGSQRT:      FLAG INDICATING A NEG SIGN UNDER A RADICAL (NO SOLN)
C  GSLAST:       VALUES OF GSSTIF AT LAST SOLUTION
C  PLAST:       VALUE OF P AT LAST SOLUTION
C  EVLAST(2):    THE LOWEST TWO EIGENVALUES AT LAST SOLUTION
C  PHELAST(NRN,2):NONDIMENSIONALIZED EIGENVECTORS AT LAST SOLUTION
C  GSSAVE(NRN,NRN): SAVED VERSION OF GSSTIF USED IN IMSL ROUTINES
C  IC:          ITERATION COUNTER
C  INEC:         COUNTER FOR # OF SOLUTIONS WITH NEG. EIGENVALUE
C  INS:         COUNTER FOR # OF TIMES CAN'T FIND A SOLUTION IN 10 ITE
C  INTERACT:     FLAG FOR INTERACTIVE OR BATCH MODE
C  IB:          FLAG FOR PASSING BIFURCATION POINT
C  IREAD:        FLAG FOR READING IN INITIAL GUESS
C ***************************************************************
C ***************************************************************
C MAIN PROGRAM
C ***************************************************************
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C     PARAMETER (NE=4,NN=19,NRN=15,NUMEI=1,IWSP=5000)
C     PARAMETER (NE=8,NN=35,NRN=31,NUMEI=1,IWSP=5000)
C     PARAMETER (NE=12,NN=51,NRN=47,NUMEI=3,IWSP=5000)
C     PARAMETER (NE=14,NN=59,NRN=55,NUMEI=1,IWSP=6237)
      PARAMETER (NE=16,NN=67,NRN=63,NUMEI=3,IWSP=9030)
C     PARAMETER (NE=20,NN=80,NRN=76,NUMEI=1,IWSP=49162)
C     PARAMETER (NE=32,NN=131,NRN=127,NUMEI=3,IWSP=34439)
C     NN=4*NE+3
      DIMENSION GU(NN),U(NRN),GUEL(NE,7),EU(NRN),CSTIF(2,2),S(NE)
      DIMENSION F(NN),QQ(NN),FF(NN)
      DIMENSION GR(NRN),GSTIF(NRN,NRN),R(NRN),REACT(4),ERR(3)
      DIMENSION GSLAST(NRN,NRN),GSSAVE(NRN,NRN)
      DIMENSION EVAL(NUMEI),EVEC(NRN,NUMEI)
      DIMENSION GULAST(NN),PHE(NRN,2),PHELAST(NRN,2),EVLAST(2)
       COMMON /WORKSP/ RWKSP
       REAL RWKSP(IWSP)
       CALL IWKIN(IWSP)
C INITIALIZE MESH CONSTANTS, FLAGS
      INEC=0
      INS=0
```

```
      IB = 0
      NPE = 7
      EVLAST(1) = 0.0
      PLAST = 0.0
C INPUT DATA
C FLAG FOR BATCH(0) OR INTERACTIVE(1)
      READ(11,*)INTERACT
C PHYSICAL CONSTANTS
      READ(11,*)EA,EI,C
      READ(11,*)SL,RL
      READ(11,*)SK,SPRL
C LOADING
      READ(11,*)Q
      READ(11,*)PINIT,PFIN,DP
C MISC INFORMATION
      READ(11,*)NSI
      READ(11,*)SFAC
C IF IREAD = 1 READ IN INITIALS U'S, IF IREAD NE 1 UINIT = 0.0
      READ(11,*)IREAD
      HEL = SL/NE
C PRINT CASE INFORMATION
      WRITE(9,1200)EA,EI,C
      WRITE(9,1201)Q
      WRITE(9,1202)PINIT
      WRITE(9,1207)DP
      WRITE(9,1206)PFIN
      WRITE(9,1203)SL,HEL,RL
      WRITE(9,1208)SK,SPRL
      WRITE(9,1204)NE,NN
      WRITE(9,1205)NSI
C INTIALIZE ITERATION COUNTER
      IC = 0
C FILL IN STIFFNESS MATRIX
      CSTIF(1,1) = EA
      CSTIF(1,2) = -EI*C
      CSTIF(2,1) = CSTIF(1,2)
      CSTIF(2,2) = EI
C ELEMENT ENDPOINTS
      DO 90 N = 1,NE
 90     S(N) = (N-1)*HEL
C INITIALIZE DISPLACEMENTS AND FORCES
      DO 100 I = 1,NN
        GU(I) = 0.D0
        F(I) = 0.D0
        QQ(I) = 0.D0
 100    FF(I) = 0.D0
      IF(IREAD.NE.1)GO TO 104
      READ(14,*)PINIT
      DO 102 I = 1,NN
 102    READ(14,*)GU(I)
C FOR DISTRIBUTED LOAD
 104  QHOT = Q*HEL/12.D0
      QQ(2) = 6.D0*QHOT
      QQ(3) = -HEL*QHOT
      DO 105 N = 2,NE
        NI = 4.D0*(N-1) + 2
```

```
 105    QQ(NI) = 12.D0*QHOT
        QQ(NN-1) = 6.D0*QHOT
        QQ(NN) = HEL*QHOT
C FOR CONCENTRATED POINT LOAD AT CENTER SPAN
        P = PINIT
C *** RETURN HERE FOR LOAD STEPPING ***********************************
 110 IC = 0
        NEGSQRT = 0
        WRITE(9,1105)
C REDUCED DISPLACEMENTS FROM GLOBAL DISPLACEMENTS
        NRNM1 = NRN-1
        DO 115 N = 1,NRNM1
 115    U(N) = GU(N + 2)
        U(NRN) = GU(NN)
        F(4*NE/2 + 2) = P
C TOTAL LOADING
        DO 120 I = 1,NN
 120    FF(I) = QQ(I) + F(I)
C *** RETURN TO HERE FOR NEXT ITERATION ******************************
 130 CONTINUE
C INITIALIZE GLOBAL STIFFNESS MATRICES
        DO 131 I = 1,NRN
          GR(I) = 0.D0
          DO 131 J = 1,NRN
 131        GSTIF(I,J) = 0.D0
C ASSIGN ELEMENT NODAL DISPLACEMENTS FROM DISPLACEMENT VECTORS GU AND U
        GUEL(1,1) = GU(1)
        GUEL(1,2) = GU(2)
        GUEL(1,3) = U(1)
        GUEL(1,7) = U(2)
        GUEL(1,4) = U(3)
        GUEL(1,5) = U(4)
        GUEL(1,6) = U(5)
        NEM1 = NE-1
        DO 135 N = 2,NEM1
          II = 4*(N-1)-1
            GUEL(N,1) = U(II)
            GUEL(N,2) = U(II + 1)
            GUEL(N,3) = U(II + 2)
            GUEL(N,7) = U(II + 3)
            GUEL(N,4) = U(II + 4)
            GUEL(N,5) = U(II + 5)
 135          GUEL(N,6) = U(II + 6)
        II = 4*(NE-1)-1
        GUEL(NE,1) = U(II)
        GUEL(NE,2) = U(II + 1)
        GUEL(NE,3) = U(II + 2)
        GUEL(NE,7) = U(II + 3)
        GUEL(NE,4) = GU(NN-2)
        GUEL(NE,5) = GU(NN-1)
        GUEL(NE,6) = U(II + 4)
C DETERMINE STIFFNESS MATRIX AND RESIDUAL
        CALL STIF(S,HEL,GUEL,C,CSTIF,NE,NRN,NSI,GSTIF,GR,REACT,NEGSQRT)
        IF(NEGSQRT.EQ.1) GO TO 700
C ADD TERMS TO STIFFNESS MATRIX FOR SPRING
```

```
      CALL SPRING(SK,SPRL,NRN,GSTIF,GR,U)
C FIND ERROR
      DO 180 I = 1,3
  180 ERR(I) = 0.D0
      NRNM1 = NRN-1
      DO 185 I = 1,NRNM1
  185   R(I) = FF(I + 2)
        R(NRN) = FF(NN)
      DO 190 I = 1,NRN
  190   R(I) = R(I)-GR(I)
      NEM1 = NE-1
      DO 195 N = 1,NEM1
        N4 = 4*N
        ERR(1) = R(N4-2)*R(N4-2) + R(N4-1)*R(N4-1) + ERR(1)
        ERR(2) = R(N4)*R(N4) + ERR(2)
  195   ERR(3) = R(N4-3)*R(N4-3) + ERR(3)
      ERR(1) = R(NRN-1)*R(NRN-1) + ERR(1)
      ERR(3) = R(NRN)*R(NRN) + R(NRN-2)*R(NRN-2) + ERR(3)
      ERR(1) = 1.D0/(2.D0*NE + 1)*DSQRT(ERR(1))
      ERR(2) = 1.D0/(NE-1)*DSQRT(ERR(2))
      ERR(3) = 1.D0/(NE + 1)*DSQRT(ERR(3))
      WRITE(9,*)(ERR(I),I = 1,3)
C IF ERROR IS SMALL U IS SOLN ... PRINT ANSWER
      IF(ERR(1).LT.1.0D-7.AND.ERR(2).LT.1.D-7.AND.ERR(3).LT.1.D-7)
     1 GO TO 500
C IF HAVE ITERATED 10 TIMES CUT LOAD STEP IN HALF
      IF(IC.GT.10)GO TO 700
C IF ERROR IS TOO LARGE ITERATE TO FIND A NEW EU TO ADD TO U
C FORM OF PROBLEM IS [GSTIF]*{EU} = {R}
C MULT GSTIF AND R BY SFAC TO AVOID OVERFLOW ERROR
      DO 215 I = 1,NRN
        R(I) = SFAC*R(I)
        DO 215 J = 1,NRN
  215     GSTIF(J,I) = SFAC*GSTIF(J,I)
C USE IMSL TO SOLVE FOR EU
      CALL DLSLSF(NRN,GSTIF,NRN,R,EU)
      DO 220 I = 1,NRN
  220   U(I) = U(I) + EU(I)
      IC = IC + 1
C LOOP BACK TO CHECK ERROR WITH NEW DISPLACEMENT VECTOR
      GO TO 130
C *** U'S HAVE BEEN SOLVED FOR ****************************************
  500 WRITE(9,1100)IC
C REGROUP GLOBAL DISPLACEMENTS
      NRNM1 = NRN-1
      DO 510 I = 1,NRNM1
  510   GU(I + 2) = U(I)
      GU(NN) = U(NRN)
C SAVE LAST GLOBAL STIFFNESS MATRIX
      DO 515 I = 1,NRN
        DO 515 J = 1,NRN
  515     GSSAVE(J,I) = GSTIF(J,I)
C COMPUTE EIGENVALUES OF TANGENT STIFFNESS MATRIX.
C     CALL DEVLSF(NRN,GSTIF,NRN,EVAL)
      CALL DEVASF(NRN,NUMEI,GSTIF,NRN,.TRUE.,EVAL)
C WRITE OUTPUT
```

```
      WRITE(9,1030)P
      WRITE(9,1110)
      WRITE(9,1120)(GU(I),I = 1,NN)
C     WRITE(9,1131)
C     WRITE(9,1120)(QQ(I),I = 1,NN)
C     WRITE(9,1132)
C     WRITE(9,1120)(F(I),I = 1,NN)
      WRITE(9,1134)
      WRITE(9,1120)(EVAL(I),I = 1,NUMEI)
C SHALLOW ARCH NONDIMENSIONALIZATION
      PNON = P*(RL*RL*RL)/(16.D0*EI)*DSQRT(EA/EI)
      WNON = GU(2*NE + 2)/(2.D0*DSQRT(EI/EA))
C  HUDDLESTONS NONDIMENSIONALIZATION
C     PNON = P*RL*RL/EI
C     WNON = GU(2*NE + 2)/RL
      WRITE(8,1040)PNON,WNON
      WRITE(15,1040)P,EVAL(1)
      IF (INTERACT.EQ.1)WRITE(6,1345)P
      IF (INTERACT.EQ.1)GO TO 800
C SAVE GLOBAL STIFFNESS MATRIX FOR LOAD STEP
 550  DO 555 I = 1,NRN
        DO 555 J = 1,NRN
 555      GSLAST(J,I) = GSSAVE(J,I)
      DO 560 I = 1,NN
 560    GULAST(I) = GU(I)
C IF EIGENVALUE  <  0 AND BIFURCATION POINT HAS NOT
C BEEN ESTABLISHED...COMPUTE EIGENVECTORS
      IF(EVAL(1).GE.0.0) GO TO 600
      PLAST = P
      IF(EVLAST(1).EQ.0)WRITE(9,1015)P
      IF(EVLAST(1).EQ.0)IB = 1
      IF(IB.GE.1)GO TO 610
      INEC = INEC + 1
      WRITE(9,1011)
      DO 570 I = 1,NRN
        DO 570 J = 1,NRN
 570      GSTIF(J,I) = GSSAVE(J,I)
      CALL DEVESF(NRN,NUMEI,GSTIF,NRN,.TRUE.,EVAL,EVEC,NRN)
      WRITE(9,1220)
      DO 580 I = 1,NRN
 580    WRITE(9,1221)(EVEC(I,J),J = 1,NUMEI)
      NNN = (NRN-1)/2
      IF(ABS(EVEC(NNN,1)).LT.1.D-8.AND.
     1ABS(EVEC(NNN + 2,1)).LT.1.D-8)GO TO 595
      IF(INEC.NE.1)GO TO 590
      PB = P-DP*EVAL(1)/(EVAL(1)-EVLAST(1))
      WRITE(9,1017)PB,P,DP,EVAL(1),EVLAST(1)
      P = PB + 3.*DP/8.
 590  P = P-DP/4.0
      GO TO 620
 595  WRITE(9,1012)P
      GO TO 610
C INCREMENT LOAD
 600  IF(INEC.LE.1)GO TO 605
      IB = IB + 1
      WRITE(9,1016)PLAST,P
```

```
      WRITE(13,1120)P,(GU(I),I = 1,NN)
  605 PLAST = P
  610 P = P + DP
  620 EVLAST(1) = EVAL(1)
C GO TO LOAD STEPPING IF WITHIN RANGE OF P
      IF(DP.GT.0.AND.P.LE.PFIN) GO TO 110
      IF(DP.LT.0.AND.P.GE.PFIN) GO TO 110
      WRITE(13,1120)PLAST,(GULAST(I),I = 1,NN)
      GO TO 900
C***IF NO SOLUTION IS FOUND ********************************************
  700 WRITE(9,1210)P,IC
C INCREMENT "NO-SOLUTION COUNTER"
      IF(INTERACT.EQ.1)WRITE(6,1210)P,IC
      IF(INTERACT.EQ.1)GO TO 830
      IF(NEGSQRT.EQ.1)GO TO 900
      INS = INS + 1
      IF(INS.GT.1)GO TO 750
      DO 725 I = 1,NRN
        DO 725 J = 1,NRN
  725       GSTIF(J,I) = GSLAST(J,I)
      CALL DEVESF(NRN,NUMEI,GSTIF,NRN,.TRUE.,EVAL,EVEC,NRN)
  750 IF(INS.GT.3)GO TO 770
      P = P-DP
      DP = DP/2.0
      DO 760 I = 1,NN
  760    GU(I) = GULAST(I)
      P = P + DP
C GO TO LOAD STEPPING
      GO TO 110
  770 WRITE(13,1120)PLAST,(GULAST(I),I = 1,NN)
      NNN = (NRN-1)/2
      IF(EVEC(NNN,1).LT.1.D-8.AND.EVEC(NNN + 2,1).LT.1.D-8)GO TO 780
      WRITE(9,1014)PLAST,P
      GO TO 900
  780 WRITE(9,1013)PLAST,P
      GO TO 900
C *** INTERACTIVE SECTION ***********************************************
  800 WRITE(6,1315)P,GU(2*NE + 2)
C COMPUTE AND NORMALIZE EIGENVECTOR
      DO 805 I = 1,NRN
        DO 805 J = 1,NRN
  805       GSTIF(J,I) = GSSAVE(J,I)
      CALL DEVESF(NRN,NUMEI,GSTIF,NRN,.TRUE.,EVAL,EVEC,NRN)
      WRITE(9,1220)
      DO 810 I = 1,NRN
  810    WRITE(9,1221)(EVEC(I,J),J = 1,NUMEI)
C NORMALIZE EIGENVECTORS
      NNN = (NRN-1)/2.
      DO 825 J = 1,2
      IF(ABS(EVEC(NNN,J)).LT.1.0E-8.AND.ABS(EVEC(NNN + 2,J)).LT.1.0E-8) TH
     1EN
        DO 815 I = 1,NRN
  815      PHE(I,J) = EVEC(I,J)/EVEC((NRN-1)/2 + 1,J)
        WRITE(6,1320)J
        ELSE
        DO 820 I = 1,NRN
```

```
820       PHE(I,J) = EVEC(I,1)/EVEC((NRN + 1)/4,J)
          WRITE(6,1325)J
        END IF
825   CONTINUE
830   WRITE(6,1310)PLAST,GULAST(2*NE + 2)
      WRITE(6,1350)EPS
      WRITE(9,1350)EPS
      WRITE(6,1330)
      READ(5,*)IUPDATE
      IF (IUPDATE.EQ.0)GO TO 850
      DO 835 I = 1,NRN
        DO 835 J = 1,NRN
835       GSLAST(J,I) = GSSAVE(J,I)
      DO 840 I = 1,NN
        DO 840 J = 1,NN
840       GULAST(I) = GU(I)
      PLAST = P
C SAVE EVAL AND PHE
      DO 845 J = 1,2
        EVLAST(J) = EVAL(J)
        DO 845 I = 1,NRN
845       PHELAST(I,J) = PHE(I,J)
850   WRITE(6,1335)
      READ(5,*)ISTOP
      IF(ISTOP.EQ.0)GO TO 890
      WRITE(6,1340)
      READ(5,*)P,EPS,NEVEC
      DO 860 I = 1,NRN-1
860   GU(I + 2) = GULAST(I + 2) + EPS*PHELAST(I,NEVEC)
      GU(NN) = GULAST(NN) + EPS*PHELAST(NRN,NEVEC)
      GO TO 110
890   WRITE(13,1120)PLAST,(GULAST(I),I = 1,NN)
C FALL THRU TO STOP
C ***********************************************************************
 1001 FORMAT(1X,'DETERMINANT = ',E15.7)
 1002 FORMAT(1X,3E8.1)
 1003 FORMAT(1X,9E7.1)
 1004 FORMAT(1X,7E10.2)
 1005 FORMAT(1X,5E10.2)
 1006 FORMAT(1X,4E10.2)
 1010 FORMAT(1X,'ERR = ',E15.7)
 1011 FORMAT(1X,'LOWEST EIGENVALUE IS NEGATIVE')
 1012 FORMAT(1X,'LOWEST EIGENVALUE IS NEGATIVE, BUT EIGENVECTOR IS SYMME
     1TRIC FOR P = ',E15.7)
 1013 FORMAT(1X,'LOWEST EIGENVECTOR IS SYMMETRIC, LIMITPOINT SUSPECTED B
     1ETWEEN P = ',E15.7,' AND P = ',E15.7)
 1014 FORMAT(1X,'LOWEST EIGENVECTOR IS ASYMMETRIC, LIMITPOINT SUSPECTED
     1BETWEEN P = ',E15.7,' AND P = ',E15.7)
 1015 FORMAT(1X,'BIFURCATION POINT SUSPECTED BEFORE P = ',E15.7)
 1016 FORMAT(1X,'BIFURCATION POINT SUSPECTED BETWEEN P = ',E15.7,' AND P
     1 = ',E15.7)
 1017 FORMAT(1X,'ESTIMATED BIFURCATION LOAD IS PB = ',E15.7/1X,4E15.7)
 1030 FORMAT(1X,'FOR P = ',E15.7)
 1040 FORMAT(1X,2E15.7)
 1100 FORMAT(1X,'AFTER ',I3,' ITERATIONS')
 1105 FORMAT(1X,'ERROR IN U,W, AND W PRIME FOR EACH ITERATION')
```

```
1110 FORMAT(1X,'GLOBAL NODAL DISPLACEMENTS')
1120 FORMAT(1X,E15.7)
1131 FORMAT(1X,'APPLIED DISTRIBUTED LOAD')
1132 FORMAT(1X,'RESULTANT NODAL FORCES')
1134 FORMAT(1X,'EIGENVALUES OF TANGENT STIFFNESS MATRIX GSTIF')
1135 FORMAT(1X,3E16.7/E16.7/3E16.7/)
1140 FORMAT(1X,'STRAINS AT NODE POINTS')
1145 FORMAT(1X,'ELEMENT',I4,5X,'EX',2X,3E15.7,/,15X,'KAPPAX',3E15.7/)
1150 FORMAT(1X,'FORCE AND MOMENT RESULTANTS AT NODE POINTS')
1155 FORMAT(1X,'ELEMENT',I4,5X,'NX',2X,3E15.7,/,17X,'MX',2X,3E15.7/)
1200 FORMAT(1X,'EA = ',E15.7,2X,'EI = ',E15.5/1X,'CURVATURE =  ',E15.7)
1201 FORMAT(1X,'MAGNITUDE OF DISTRIBUTED LOAD = ',E15.7)
1202 FORMAT(1X,'MAGNITUDE OF INITIAL CENTER LOAD = ',E15.7)
1203 FORMAT(1X,'ARCH LENGTH = ',E15.7/1X,'ELEMENT LENGTH = ',E15.7,/
   11X,'SPAN LENGTH = ',E15.7)
1204 FORMAT(1X,'NUMBER OF ELEMENTS = ',I4/1X,'NUMBER OF NODES = ',I4)
1205 FORMAT(1X,'NUMBER OF SIMPSONS RULE APPLICATION IN STFINT = ',I5/)
1206 FORMAT(1X,'MAGNITUDE OF FINAL CENTER LOAD = ',E15.7)
1207 FORMAT(1X,'MAGNITUDE OF CENTER LOAD INCREMENT= ',E15.7)
1208 FORMAT(1X,'SPRING STIFFNESS = ',E15.7,'  INITIAL LENGTH = ',E15.7)
1209 FORMAT(1X,'EPSILON = ',E15.7)
1210 FORMAT(1X,'FOR P = ',E15.7/1X,'NO SOLUTION FOUND AFTER ',I4,' ITER
   1ATIONS')
1211 FORMAT(1X,'TRYING TO GO OVER LIMIT POINT')
1215 FORMAT(1X,I5)
1216 FORMAT(1X,E20.12)
1220 FORMAT(1X,'EIGENVECTORS ASSOCIATED WITH LOWEST EIGENVALUES')
1221 FORMAT(1X,3E17.7)
1310 FORMAT(1X,'LAST SOLN HAS P = ',E15.7,'  W(0) = ',E15.7)
1315 FORMAT(1X,'CURRENT SOLN HAS P = ',E15.7,'  W(0) = ',E15.7)
1320 FORMAT(1X,'EIGENVECTOR',I2,' IS SYMMETRIC')
1325 FORMAT(1X,'EIGENVECTOR',I2,' IS ASYMMETRIC')
1330 FORMAT(1X,'DO YOU WANT TO UPDATE PLAST, EVLAST, GULAST, AND PHELAS
   1T? (0=NO, 1=YES)'/1X,'(IF FIRST TIME THROUGH SAY YES)')
1335 FORMAT(1X,'DO YOU WANT TO GUESS AT P AND EPS? OR STOP?'/1X,'(0=STO
   1P, 1=GUESS)')
1340 FORMAT(1X,'ENTER P AND EPS AND WHICH EIGENVECTOR TO USE')
1345 FORMAT(1X,'SOLUTION FOUND FOR P =',E15.7)
1350 FORMAT(1X,'VALUE OF EPS =',E15.7)
C ***********************************************************************
 900 STOP
     END
C ***********************************************************************
C SUBPROGRAM TO ASSEMBLE THE GLOBAL STIFFNESS MATRIX
C ***********************************************************************
     SUBROUTINE STIF(S,HEL,GUEL,C,CSTIF,NE,NRN,NSI,GSTIF,GR,REACT,NEGSQ
    1RT)
     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
     DIMENSION S(NE),GUEL(NE,7),CSTIF(2,2)
     DIMENSION GSTIF(NRN,NRN),GR(NRN),REACT(4)
     DIMENSION ELSTIF(7,7),ELR(7),UEL(7)
C FILL IN GLOBAL RESIDUAL STIFFNESS
C CONTRIBUTIONS FROM FIRST ELEMENT
     N=1
     DO 10 I=1,7
 10   UEL(I)=GUEL(1,I)
```

```
      SEL = S(N)
      CALL STFINT(SEL,HEL,UEL,NSI,C,CSTIF,ELSTIF,ELR,NEGSQRT)
      IF(NEGSQRT.EQ.1)GO TO 250
      GSTIF(1,1) = ELSTIF(3,3)
      GSTIF(1,2) = ELSTIF(3,7)
      GSTIF(1,3) = ELSTIF(3,4)
      GSTIF(1,4) = ELSTIF(3,5)
      GSTIF(1,5) = ELSTIF(3,6)
      GR(1) = ELR(3)
      GSTIF(2,1) = ELSTIF(7,3)
      GSTIF(2,2) = ELSTIF(7,7)
      GSTIF(2,3) = ELSTIF(7,4)
      GSTIF(2,4) = ELSTIF(7,5)
      GSTIF(2,5) = ELSTIF(7,6)
      GR(2) = ELR(7)
      DO 150 I = 3,5
      II = I + 1
        GSTIF(I,1) = ELSTIF(II,3)
        GSTIF(I,2) = ELSTIF(II,7)
        GSTIF(I,3) = ELSTIF(II,4)
        GSTIF(I,4) = ELSTIF(II,5)
        GSTIF(I,5) = ELSTIF(II,6)
 150    GR(I) = ELR(II)
      REACT(1) = ELR(1)
      REACT(2) = ELR(2)
C CONTRIBUTION FROM INTERNAL ELEMENTS
      NEM1 = NE-1
      DO 165 N = 2,NEM1
        DO 156 I = 1,7
 156      UEL(I) = GUEL(N,I)
        SEL = S(N)
        CALL STFINT(SEL,HEL,UEL,NSI,C,CSTIF,ELSTIF,ELR,NEGSQRT)
      IF(NEGSQRT.EQ.1)GO TO 250
        K = 4*(N-1)-1
        DO 155 I = 1,3
          L = K + I-1
          GSTIF(L,K) = ELSTIF(I,1) + GSTIF(L,K)
          GSTIF(L,K + 1) = ELSTIF(I,2) + GSTIF(L,K + 1)
          GSTIF(L,K + 2) = ELSTIF(I,3) + GSTIF(L,K + 2)
          GSTIF(L,K + 3) = ELSTIF(I,7)
          GSTIF(L,K + 4) = ELSTIF(I,4)
          GSTIF(L,K + 5) = ELSTIF(I,5)
          GSTIF(L,K + 6) = ELSTIF(I,6)
 155      GR(L) = ELR(I) + GR(L)
        L = K + 3
          GSTIF(L,K) = ELSTIF(7,1)
          GSTIF(L,K + 1) = ELSTIF(7,2)
          GSTIF(L,K + 2) = ELSTIF(7,3)
          GSTIF(L,K + 3) = ELSTIF(7,7)
          GSTIF(L,K + 4) = ELSTIF(7,4)
          GSTIF(L,K + 5) = ELSTIF(7,5)
          GSTIF(L,K + 6) = ELSTIF(7,6)
          GR(L) = ELR(7)
        DO 160 I = 4,6
          L = K + I
          GSTIF(L,K) = ELSTIF(I,1)
```

```
          GSTIF(L,K + 1) = ELSTIF(I,2)
          GSTIF(L,K + 2) = ELSTIF(I,3)
          GSTIF(L,K + 3) = ELSTIF(I,7)
          GSTIF(L,K + 4) = ELSTIF(I,4)
          GSTIF(L,K + 5) = ELSTIF(I,5)
          GSTIF(L,K + 6) = ELSTIF(I,6)
160         GR(L) = ELR(I)
165   CONTINUE
C CONTRIBUTION FROM LAST ELEMENT
        DO 166 I = 1,7
166         UEL(I) = GUEL(NE,I)
        SEL = S(NE)
        CALL STFINT(SEL,HEL,UEL,NSI,C,CSTIF,ELSTIF,ELR,NEGSQRT)
      IF(NEGSQRT.EQ.1)GO TO 250
      K = 4*(NE-1)-1
      DO 170 I = 1,3
        L = K + (I-1)
          GSTIF(L,K) = ELSTIF(I,1) + GSTIF(L,K)
          GSTIF(L,K + 1) = ELSTIF(I,2) + GSTIF(L,K + 1)
          GSTIF(L,K + 2) = ELSTIF(I,3) + GSTIF(L,K + 2)
          GSTIF(L,K + 3) = ELSTIF(I,7)
          GSTIF(L,K + 4) = ELSTIF(I,6)
170       GR(L) = ELR(I) + GR(L)
        L = K + 3
          GSTIF(L,K) = ELSTIF(7,1)
          GSTIF(L,K + 1) = ELSTIF(7,2)
          GSTIF(L,K + 2) = ELSTIF(7,3)
          GSTIF(L,K + 3) = ELSTIF(7,7)
          GSTIF(L,K + 4) = ELSTIF(7,6)
          GR(L) = ELR(7)
        L = K + 4
          GSTIF(L,K) = ELSTIF(6,1)
          GSTIF(L,K + 1) = ELSTIF(6,2)
          GSTIF(L,K + 2) = ELSTIF(6,3)
          GSTIF(L,K + 3) = ELSTIF(6,7)
          GSTIF(L,K + 4) = ELSTIF(6,6)
          GR(L) = ELR(6)
        REACT(3) = ELR(5)
        REACT(4) = ELR(6)
250   RETURN
      END
C *********************************************************************
C SUBPROGRAM TO INTEGRATE ELEMENT STIFFNESS TERMS
C *********************************************************************
      SUBROUTINE STFINT(S0,HEL,UEL,NSI,C,CSTIF,ELSTIF,ELR,NEGSQRT)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION UEL(7),CSTIF(2,2)
      DIMENSION ELSTIF(7,7),ELR(7)
      DIMENSION ELK1(7,7),ELK2(7,7),ELKA(7,7),ELKB(7,7),ELK(7,7)
      DIMENSION ELKG1(7,7),ELKG2(7,7),ELKGA(7,7),ELKGB(7,7),ELKG(7,7)
      DIMENSION SENDK(7,7),SENDKG(7,7),SMIDK(7,7),SMIDKG(7,7)
      DIMENSION ELR1(7),ELR2(7),ELRA(7),ELRB(7)
      DIMENSION SENDR(7),SMIDR(7)
      SN = S0 + HEL
      TWOHI = HEL/NSI
```

```
      HI = TWOHI/2.D0
C*  SET SUMS TO ZERO
      DO 50 I = 1,7
        SENDR(I) = 0.D0
        SMIDR(I) = 0.D0
        DO 50 J = 1,7
          SENDK(I,J) = 0.D0
          SMIDK(I,J) = 0.D0
          SENDKG(I,J) = 0.D0
 50        SMIDKG(I,J) = 0.D0
C   BEGIN INTEGRATION LOOP
      DO 100 K = 1,NSI
      S1 = S0 + (K-1)*TWOHI
      S2 = S1 + HI
      CALL ELINTS(S1,S0,HEL,C,UEL,CSTIF,ELK1,ELKG1,ELR1,NEGSQRT)
      IF(NEGSQRT.EQ.1)GO TO 250
      CALL ELINTS(S2,S0,HEL,C,UEL,CSTIF,ELK2,ELKG2,ELR2,NEGSQRT)
      IF(NEGSQRT.EQ.1)GO TO 250
C**  COMPUTE SUMS
      DO 100 I = 1,7
        SENDR(I) = SENDR(I) + ELR1(I)
        SMIDR(I) = SMIDR(I) + ELR2(I)
        DO 100 J = 1,7
        SENDK(I,J) = SENDK(I,J) + ELK1(I,J)
        SMIDK(I,J) = SMIDK(I,J) + ELK2(I,J)
        SENDKG(I,J) = SENDKG(I,J) + ELKG1(I,J)
 100    SMIDKG(I,J) = SMIDKG(I,J) + ELKG2(I,J)
      CALL ELINTS(S0,S0,HEL,C,UEL,CSTIF,ELKA,ELKGA,ELRA,NEGSQRT)
      IF(NEGSQRT.EQ.1)GO TO 250
      CALL ELINTS(SN,S0,HEL,C,UEL,CSTIF,ELKB,ELKGB,ELRB,NEGSQRT)
      IF(NEGSQRT.EQ.1)GO TO 250
      DO 200 I = 1,7
      ELR(I) = (2.D0*SENDR(I) + 4.D0*SMIDR(I)-ELRA(I) + ELRB(I))*HI/3.D0
        DO 200 J = 1,7
      ELK(I,J) = (2.D0*SENDK(I,J) + 4.D0*SMIDK(I,J)-ELKA(I,J) + ELKB(I,J))*HI
     1/3.D0
 200  ELKG(I,J) = (2.D0*SENDKG(I,J) + 4.D0*SMIDKG(I,J)-ELKGA(I,J) + ELKGB(I,J
     1))*HI/3.D0
       CALL MATADD(1.D0,ELK,1.D0,ELKG,ELSTIF,7,7)
 250   RETURN
       END
C ***********************************************************************
C PROGRAM TO COMPUTE VALUES OF PARTS OF ELE MATRICES AT INT POINTS.
C ***********************************************************************
      SUBROUTINE ELINTS(S,S0,HEL,C,UEL,CSTIF,ELKI,ELKGI,ELRI,NEGSQRT)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION UEL(7),CSTIF(2,2)
      DIMENSION ELKI(7,7),ELKGI(7,7),ELRI(7)
      DIMENSION ZA1(7),ZA2(7),ZB1(7),ZB2(7),ZUD1(7),ZUD2(7),ZUE2(7)
      DIMENSION ZD1(7,7),ZD2(7,7),ZE1(7,7),ZE2(7,7)
      DIMENSION A1(7),A2(7),B1(7),B2(7),A(2,7),B(2,7),D(7,7),E(7,7)
      DIMENSION BTC(7,2),CA(2,7),FORC(2),DUMMY(7),DUMMYS(7,7)
C**  COMPUTE ...
      CALL MATXA1(S,S0,HEL,C,ZA1)
      CALL MATXD1(S,S0,HEL,C,ZD1)
      CALL MATXD2(S,S0,HEL,C,ZD2)
```

```
      CALL MATXA2(S,S0,HEL,C,ZA2)
      CALL MATXB2(S,S0,HEL,C,ZB2)
      CALL MATXE1(S,S0,HEL,C,ZE1)
      CALL MATXE2(S,S0,HEL,C,ZE2)
      CALL CMMULT(UEL,ZD1,ZUD1,7,7)
      CALL CMMULT(UEL,ZD2,ZUD2,7,7)
      CALL COLMULT(ZUD2,UEL,GAMN2,7)
      CALL CMMULT(UEL,ZE2,ZUE2,7,7)
      CALL COLMULT(ZUE2,UEL,GAMNNP,7)
      CALL COLMULT(ZA1,UEL,GAMT,7)
      CALL COLMULT(ZB2,UEL,GAMN,7)
      CALL COLMULT(ZA2,UEL,GAMNP,7)
      GAMT2 = GAMT*GAMT
      EPS0 = GAMT + 0.5*GAMT2 + 0.5*GAMN2
      IF((1.-GAMN2).LT.0)GO TO 100
      BETAP = GAMNP/(DSQRT(1-GAMN2))
C** FORM MATRICES
C** A1
      CALL CADD(1.D0,ZA1,0.5D0,ZUD1,DUMMY,7)
      CALL CADD(1.D0,DUMMY,0.5D0,ZUD2,A1,7)
C** A2
      FAC = 1.D0/DSQRT(1-GAMN2)
      CALL SMULT(FAC,ZA2,A2,7)
C** B1
      CALL CADD(1.D0,ZA1,1.D0,ZUD1,DUMMY,7)
      CALL CADD(1.D0,DUMMY,1.D0,ZUD2,B1,7)
C** B2
      FAC = GAMNNP/(1.D0-GAMN2)**1.5
      CALL CADD(1.D0,A2,FAC,ZB2,B2,7)
C** D
      CALL MATADD(1.D0,ZD1,1.D0,ZD2,D,7,7)
C** E
      CALL MATADD(1.D0,ZE1,1.D0,ZE2,DUMMYS,7,7)
      FAC = (1.D0-GAMN2)**1.5
      FACA = GAMN/FAC
      FACB = GAMNP*(1.D0 + 2.D0*GAMN2)/((1.D0-GAMN2)**2.5)
      CALL MATADD(FACA,DUMMYS,FACB,ZD2,E,7,7)
C** A AND B
      DO 50 I = 1,7
        A(1,I) = A1(I)
        B(1,I) = B1(I)
        A(2,I) = A2(I)
50      B(2,I) = B2(I)
C** ELSTIF(ELK)
      CALL MATTMULT(B,CSTIF,BTC,7,2,2)
      CALL MATMULT(BTC,B,ELKI,7,2,7)
      CALL MATMULT(CSTIF,A,CA,2,2,7)
      CALL MCMULT(CA,UEL,FORC,2,7)
      CALL MATADD(FORC(1),D,FORC(2),E,ELKGI,7,7)
      CALL CADD(FORC(1),B1,FORC(2),B2,ELRI,7)
      GO TO 150
100   WRITE(6,1000)
      WRITE(9,1000)
      NEGSQRT = 1
150   RETURN
1000  FORMAT(1X,'CANNOT FILL IN STIFFNESS MATRIX, ARG OF SQRT < 0')
```

```
      END
C ***********************************************************************
C CODE FOR MATRIX A1
C ***********************************************************************
      SUBROUTINE MATXA1(S,S1,H,C,ZATA1)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION ZATA1(7)
      ZATA1(1) = 2*(2*(S-S1)/H-1)/H-1/H
      ZATA1(2) = -C*(2*(S-S1)**3/H**3-3*(S-S1)**2/H**2+1)
      ZATA1(3) = C*(1-(S-S1)/H)**2*(S-S1)
      ZATA1(4) = 2*(2*(S-S1)/H-1)/H+1/H
      ZATA1(5) = -C*(3*(S-S1)**2/H**2-2*(S-S1)**3/H**3)
      ZATA1(6) = C*((S-S1)**2/H**2-(S-S1)/H)*(S-S1)
      ZATA1(7) = -4*(2*(S-S1)/H-1)/H
      RETURN
      END
C ***********************************************************************
C CODE FOR MATRIX A2
C ***********************************************************************
      SUBROUTINE MATXA2(S,S1,H,C,ZATA2)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION ZATA2(7)
      ZATA2(1) = C*(2*(2*(S-S1)/H-1)/H-1/H)
      ZATA2(2) = 12*(S-S1)/H**3-6/H**2
      ZATA2(3) = 4*(1-(S-S1)/H)/H-2*(S-S1)/H**2
      ZATA2(4) = C*(2*(2*(S-S1)/H-1)/H+1/H)
      ZATA2(5) = 6/H**2-12*(S-S1)/H**3
      ZATA2(6) = 2/H-6*(S-S1)/H**2
      ZATA2(7) = -4*C*(2*(S-S1)/H-1)/H
      RETURN
      END
C ***********************************************************************
C CODE FOR MATRIX B2
C ***********************************************************************
      SUBROUTINE MATXB2(S,S1,H,C,ZATB2)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION ZATB2(7)
      ZATB2(1) = C*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)
      ZATB2(2) = 6*(S-S1)**2/H**3-6*(S-S1)/H**2
      ZATB2(3) = 2*(1-(S-S1)/H)*(S-S1)/H-(1-(S-S1)/H)**2
      ZATB2(4) = C*((2*(S-S1)/H-1)**2/2.0+(2*(S-S1)/H-1)/2.0)
      ZATB2(5) = 6*(S-S1)/H**2-6*(S-S1)**2/H**3
      ZATB2(6) = -(S-S1)**2/H**2-(2*(S-S1)/H**2-1/H)*(S-S1)+(S-S1)/H
      ZATB2(7) = C*(1-(2*(S-S1)/H-1)**2)
      RETURN
      END
C ***********************************************************************
C CODE FOR MATRIX D1
C ***********************************************************************
      SUBROUTINE MATXD1(S,S1,H,C,ZATD1)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION ZATD1(7,7)
      ZATD1(1,1) = (2*(2*(S-S1)/H-1)/H-1/H)**2
      ZATD1(1,2) = -C*(2*(2*(S-S1)/H-1)/H-1/H)*(2*(S-S1)**3/H**3-3*(S-S1
     1   )**2/H**2+1)
      ZATD1(1,3) = C*(2*(2*(S-S1)/H-1)/H-1/H)*(1-(S-S1)/H)**2*(S-S1)
```

```
ZATD1(1,4) = (2*(2*(S-S1)/H-1)/H-1/H)*(2*(2*(S-S1)/H-1)/H + 1/H)
ZATD1(1,5) = -C*(2*(2*(S-S1)/H-1)/H-1/H)*(3*(S-S1)**2/H**2-2*(S-S1
1  )**3/H**3)
ZATD1(1,6) = C*(2*(2*(S-S1)/H-1)/H-1/H)*((S-S1)**2/H**2-(S-S1)/H)*
1  (S-S1)
ZATD1(1,7) = -4*(2*(2*(S-S1)/H-1)/H-1/H)*(2*(S-S1)/H-1)/H
ZATD1(2,1) = -C*(2*(2*(S-S1)/H-1)/H-1/H)*(2*(S-S1)**3/H**3-3*(S-S1
1  )**2/H**2 + 1)
ZATD1(2,2) = C**2*(2*(S-S1)**3/H**3-3*(S-S1)**2/H**2 + 1)**2
ZATD1(2,3) = -C**2*(1-(S-S1)/H)**2*(2*(S-S1)**3/H**3-3*(S-S1)**2/H
1  **2 + 1)*(S-S1)
ZATD1(2,4) = -C*(2*(2*(S-S1)/H-1)/H + 1/H)*(2*(S-S1)**3/H**3-3*(S-S1
1  )**2/H**2 + 1)
ZATD1(2,5) = C**2*(3*(S-S1)**2/H**2-2*(S-S1)**3/H**3)*(2*(S-S1)**3
1  /H**3-3*(S-S1)**2/H**2 + 1)
ZATD1(2,6) = -C**2*((S-S1)**2/H**2-(S-S1)/H)*(2*(S-S1)**3/H**3-3*(
1  S-S1)**2/H**2 + 1)*(S-S1)
ZATD1(2,7) = 4*C*(2*(S-S1)/H-1)*(2*(S-S1)**3/H**3-3*(S-S1)**2/H**2
1  + 1)/H
ZATD1(3,1) = C*(2*(2*(S-S1)/H-1)/H-1/H)*(1-(S-S1)/H)**2*(S-S1)
ZATD1(3,2) = -C**2*(1-(S-S1)/H)**2*(2*(S-S1)**3/H**3-3*(S-S1)**2/H
1  **2 + 1)*(S-S1)
ZATD1(3,3) = C**2*(1-(S-S1)/H)**4*(S-S1)**2
ZATD1(3,4) = C*(2*(2*(S-S1)/H-1)/H + 1/H)*(1-(S-S1)/H)**2*(S-S1)
ZATD1(3,5) = -C**2*(1-(S-S1)/H)**2*(3*(S-S1)**2/H**2-2*(S-S1)**3/H
1  **3)*(S-S1)
ZATD1(3,6) = C**2*(1-(S-S1)/H)**2*((S-S1)**2/H**2-(S-S1)/H)*(S-S1)
1  **2
ZATD1(3,7) = -4*C*(1-(S-S1)/H)**2*(2*(S-S1)/H-1)*(S-S1)/H
ZATD1(4,1) = (2*(2*(S-S1)/H-1)/H-1/H)*(2*(2*(S-S1)/H-1)/H + 1/H)
ZATD1(4,2) = -C*(2*(2*(S-S1)/H-1)/H + 1/H)*(2*(S-S1)**3/H**3-3*(S-S1
1  )**2/H**2 + 1)
ZATD1(4,3) = C*(2*(2*(S-S1)/H-1)/H + 1/H)*(1-(S-S1)/H)**2*(S-S1)
ZATD1(4,4) = (2*(2*(S-S1)/H-1)/H + 1/H)**2
ZATD1(4,5) = -C*(2*(2*(S-S1)/H-1)/H + 1/H)*(3*(S-S1)**2/H**2-2*(S-S1
1  )**3/H**3)
ZATD1(4,6) = C*(2*(2*(S-S1)/H-1)/H + 1/H)*((S-S1)**2/H**2-(S-S1)/H)*
1  (S-S1)
ZATD1(4,7) = -4*(2*(2*(S-S1)/H-1)/H + 1/H)*(2*(S-S1)/H-1)/H
ZATD1(5,1) = -C*(2*(2*(S-S1)/H-1)/H-1/H)*(3*(S-S1)**2/H**2-2*(S-S1
1  )**3/H**3)
ZATD1(5,2) = C**2*(3*(S-S1)**2/H**2-2*(S-S1)**3/H**3)*(2*(S-S1)**3
1  /H**3-3*(S-S1)**2/H**2 + 1)
ZATD1(5,3) = -C**2*(1-(S-S1)/H)**2*(3*(S-S1)**2/H**2-2*(S-S1)**3/H
1  **3)*(S-S1)
ZATD1(5,4) = -C*(2*(2*(S-S1)/H-1)/H + 1/H)*(3*(S-S1)**2/H**2-2*(S-S1
1  )**3/H**3)
ZATD1(5,5) = C**2*(3*(S-S1)**2/H**2-2*(S-S1)**3/H**3)**2
ZATD1(5,6) = -C**2*((S-S1)**2/H**2-(S-S1)/H)*(3*(S-S1)**2/H**2-2*(
1  S-S1)**3/H**3)*(S-S1)
ZATD1(5,7) = 4*C*(2*(S-S1)/H-1)*(3*(S-S1)**2/H**2-2*(S-S1)**3/H**3
1  )/H
ZATD1(6,1) = C*(2*(2*(S-S1)/H-1)/H-1/H)*((S-S1)**2/H**2-(S-S1)/H)*
1  (S-S1)
ZATD1(6,2) = -C**2*((S-S1)**2/H**2-(S-S1)/H)*(2*(S-S1)**3/H**3-3*(
1  S-S1)**2/H**2 + 1)*(S-S1)
```

```
      ZATD1(6,3) = C**2*(1-(S-S1)/H)**2*((S-S1)**2/H**2-(S-S1)/H)*(S-S1)
     1  **2
      ZATD1(6,4) = C*(2*(2*(S-S1)/H-1)/H + 1/H)*((S-S1)**2/H**2-(S-S1)/H)*
     1  (S-S1)
      ZATD1(6,5) = -C**2*((S-S1)**2/H**2-(S-S1)/H)*(3*(S-S1)**2/H**2-2*(
     1  S-S1)**3/H**3)*(S-S1)
      ZATD1(6,6) = C**2*((S-S1)**2/H**2-(S-S1)/H)**2*(S-S1)**2
      ZATD1(6,7) = -4*C*(2*(S-S1)/H-1)*((S-S1)**2/H**2-(S-S1)/H)*(S-S1)/
     1  H
      ZATD1(7,1) = -4*(2*(2*(S-S1)/H-1)/H-1/H)*(2*(S-S1)/H-1)/H
      ZATD1(7,2) = 4*C*(2*(S-S1)/H-1)*(2*(S-S1)**3/H**3-3*(S-S1)**2/H**2
     1  + 1)/H
      ZATD1(7,3) = -4*C*(1-(S-S1)/H)**2*(2*(S-S1)/H-1)*(S-S1)/H
      ZATD1(7,4) = -4*(2*(2*(S-S1)/H-1)/H + 1/H)*(2*(S-S1)/H-1)/H
      ZATD1(7,5) = 4*C*(2*(S-S1)/H-1)*(3*(S-S1)**2/H**2-2*(S-S1)**3/H**3
     1  )/H
      ZATD1(7,6) = -4*C*(2*(S-S1)/H-1)*((S-S1)**2/H**2-(S-S1)/H)*(S-S1)/
     1  H
      ZATD1(7,7) = 16*(2*(S-S1)/H-1)**2/H**2
      RETURN
      END
C ***********************************************************************
C CODE FOR MATRIX D2
C ***********************************************************************
      SUBROUTINE MATXD2(S,S1,H,C,ZATD2)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION ZATD2(7,7)
      ZATD2(1,1) = C**2*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)**2
      ZATD2(1,2) = C*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)*(6*(S-S1
     1  )**2/H**3-6*(S-S1)/H**2)
      ZATD2(1,3) = C*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)*(2*(1-(S
     1  -S1)/H)*(S-S1)/H-(1-(S-S1)/H)**2)
      ZATD2(1,4) = C**2*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)*((2*(
     1  S-S1)/H-1)**2/2.0 + (2*(S-S1)/H-1)/2.0)
      ZATD2(1,5) = C*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)*(6*(S-S1
     1  )/H**2-6*(S-S1)**2/H**3)
      ZATD2(1,6) = C*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)*(-(S-S1)
     1  **2/H**2-(2*(S-S1)/H**2-1/H)*(S-S1) + (S-S1)/H)
      ZATD2(1,7) = C**2*(1-(2*(S-S1)/H-1)**2)*((2*(S-S1)/H-1)**2/2.0-(2*
     1  (S-S1)/H-1)/2.0)
      ZATD2(2,1) = C*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)*(6*(S-S1
     1  )**2/H**3-6*(S-S1)/H**2)
      ZATD2(2,2) = (6*(S-S1)**2/H**3-6*(S-S1)/H**2)**2
      ZATD2(2,3) = (2*(1-(S-S1)/H)*(S-S1)/H-(1-(S-S1)/H)**2)*(6*(S-S1)**
     1  2/H**3-6*(S-S1)/H**2)
      ZATD2(2,4) = C*((2*(S-S1)/H-1)**2/2.0 + (2*(S-S1)/H-1)/2.0)*(6*(S-S1
     1  )**2/H**3-6*(S-S1)/H**2)
      ZATD2(2,5) = (6*(S-S1)/H**2-6*(S-S1)**2/H**3)*(6*(S-S1)**2/H**3-6*
     1  (S-S1)/H**2)
      ZATD2(2,6) = (6*(S-S1)**2/H**3-6*(S-S1)/H**2)*(-(S-S1)**2/H**2-(2*
     1  (S-S1)/H**2-1/H)*(S-S1) + (S-S1)/H)
      ZATD2(2,7) = C*(1-(2*(S-S1)/H-1)**2)*(6*(S-S1)**2/H**3-6*(S-S1)/H*
     1  *2)
      ZATD2(3,1) = C*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)*(2*(1-(S
     1  -S1)/H)*(S-S1)/H-(1-(S-S1)/H)**2)
      ZATD2(3,2) = (2*(1-(S-S1)/H)*(S-S1)/H-(1-(S-S1)/H)**2)*(6*(S-S1)**
```

```
1   2/H**3-6*(S-S1)/H**2)
 ZATD2(3,3) = (2*(1-(S-S1)/H)*(S-S1)/H-(1-(S-S1)/H)**2)**2
 ZATD2(3,4) = C*((2*(S-S1)/H-1)**2/2.0+(2*(S-S1)/H-1)/2.0)*(2*(1-(S
1   -S1)/H)*(S-S1)/H-(1-(S-S1)/H)**2)
 ZATD2(3,5) = (2*(1-(S-S1)/H)*(S-S1)/H-(1-(S-S1)/H)**2)*(6*(S-S1)/H
1   **2-6*(S-S1)**2/H**3)
 ZATD2(3,6) = (2*(1-(S-S1)/H)*(S-S1)/H-(1-(S-S1)/H)**2)*(-(S-S1)**2
1   /H**2-(2*(S-S1)/H**2-1/H)*(S-S1)+(S-S1)/H)
 ZATD2(3,7) = C*(1-(2*(S-S1)/H-1)**2)*(2*(1-(S-S1)/H)*(S-S1)/H-(1-(
1   S-S1)/H)**2)
 ZATD2(4,1) = C**2*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)*((2*(
1   S-S1)/H-1)**2/2.0+(2*(S-S1)/H-1)/2.0)
 ZATD2(4,2) = C*((2*(S-S1)/H-1)**2/2.0+(2*(S-S1)/H-1)/2.0)*(6*(S-S1
1   )**2/H**3-6*(S-S1)/H**2)
 ZATD2(4,3) = C*((2*(S-S1)/H-1)**2/2.0+(2*(S-S1)/H-1)/2.0)*(2*(1-(S
1   -S1)/H)*(S-S1)/H-(1-(S-S1)/H)**2)
 ZATD2(4,4) = C**2*((2*(S-S1)/H-1)**2/2.0+(2*(S-S1)/H-1)/2.0)**2
 ZATD2(4,5) = C*((2*(S-S1)/H-1)**2/2.0+(2*(S-S1)/H-1)/2.0)*(6*(S-S1
1   )/H**2-6*(S-S1)**2/H**3)
 ZATD2(4,6) = C*((2*(S-S1)/H-1)**2/2.0+(2*(S-S1)/H-1)/2.0)*(-(S-S1)
1   **2/H**2-(2*(S-S1)/H**2-1/H)*(S-S1)+(S-S1)/H)
 ZATD2(4,7) = C**2*(1-(2*(S-S1)/H-1)**2)*((2*(S-S1)/H-1)**2/2.0+(2*
1   (S-S1)/H-1)/2.0)
 ZATD2(5,1) = C*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)*(6*(S-S1
1   )/H**2-6*(S-S1)**2/H**3)
 ZATD2(5,2) = (6*(S-S1)/H**2-6*(S-S1)**2/H**3)*(6*(S-S1)**2/H**3-6*
1   (S-S1)/H**2)
 ZATD2(5,3) = (2*(1-(S-S1)/H)*(S-S1)/H-(1-(S-S1)/H)**2)*(6*(S-S1)/H
1   **2-6*(S-S1)**2/H**3)
 ZATD2(5,4) = C*((2*(S-S1)/H-1)**2/2.0+(2*(S-S1)/H-1)/2.0)*(6*(S-S1
1   )/H**2-6*(S-S1)**2/H**3)
 ZATD2(5,5) = (6*(S-S1)/H**2-6*(S-S1)**2/H**3)**2
 ZATD2(5,6) = (6*(S-S1)/H**2-6*(S-S1)**2/H**3)*(-(S-S1)**2/H**2-(2*
1   (S-S1)/H**2-1/H)*(S-S1)+(S-S1)/H)
 ZATD2(5,7) = C*(1-(2*(S-S1)/H-1)**2)*(6*(S-S1)/H**2-6*(S-S1)**2/H*
1   *3)
 ZATD2(6,1) = C*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)*(-(S-S1)
1   **2/H**2-(2*(S-S1)/H**2-1/H)*(S-S1)+(S-S1)/H)
 ZATD2(6,2) = (6*(S-S1)**2/H**3-6*(S-S1)/H**2)*(-(S-S1)**2/H**2-(2*
1   (S-S1)/H**2-1/H)*(S-S1)+(S-S1)/H)
 ZATD2(6,3) = (2*(1-(S-S1)/H)*(S-S1)/H-(1-(S-S1)/H)**2)*(-(S-S1)**2
1   /H**2-(2*(S-S1)/H**2-1/H)*(S-S1)+(S-S1)/H)
 ZATD2(6,4) = C*((2*(S-S1)/H-1)**2/2.0+(2*(S-S1)/H-1)/2.0)*(-(S-S1)
1   **2/H**2-(2*(S-S1)/H**2-1/H)*(S-S1)+(S-S1)/H)
 ZATD2(6,5) = (6*(S-S1)/H**2-6*(S-S1)**2/H**3)*(-(S-S1)**2/H**2-(2*
1   (S-S1)/H**2-1/H)*(S-S1)+(S-S1)/H)
 ZATD2(6,6) = (-(S-S1)**2/H**2-(2*(S-S1)/H**2-1/H)*(S-S1)+(S-S1)/H)
1   **2
 ZATD2(6,7) = C*(1-(2*(S-S1)/H-1)**2)*(-(S-S1)**2/H**2-(2*(S-S1)/H*
1   *2-1/H)*(S-S1)+(S-S1)/H)
 ZATD2(7,1) = C**2*(1-(2*(S-S1)/H-1)**2)*((2*(S-S1)/H-1)**2/2.0-(2*
1   (S-S1)/H-1)/2.0)
 ZATD2(7,2) = C*(1-(2*(S-S1)/H-1)**2)*(6*(S-S1)**2/H**3-6*(S-S1)/H*
1   *2)
 ZATD2(7,3) = C*(1-(2*(S-S1)/H-1)**2)*(2*(1-(S-S1)/H)*(S-S1)/H-(1-(
1   S-S1)/H)**2)
```

```
      ZATD2(7,4) = C**2*(1-(2*(S-S1)/H-1)**2)*((2*(S-S1)/H-1)**2/2.0 + (2*
     1  (S-S1)/H-1)/2.0)
      ZATD2(7,5) = C*(1-(2*(S-S1)/H-1)**2)*(6*(S-S1)/H**2-6*(S-S1)**2/H*
     1  *3)
      ZATD2(7,6) = C*(1-(2*(S-S1)/H-1)**2)*(-(S-S1)**2/H**2-(2*(S-S1)/H*
     1  *2-1/H)*(S-S1) + (S-S1)/H)
      ZATD2(7,7) = C**2*(1-(2*(S-S1)/H-1)**2)**2
      RETURN
      END
C ***********************************************************************
C CODE FOR MATRIX D
C ***********************************************************************
      SUBROUTINE MATXD(S,S1,H,C,ZATD)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION ZATD(7,7)
      ZATD(1,1) = C**2*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)**2 + (2*
     1  (2*(S-S1)/H-1)/H-1/H)**2
      ZATD(1,2) = C*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)*(6*(S-S1)
     1  **2/H**3-6*(S-S1)/H**2)-C*(2*(2*(S-S1)/H-1)/H-1/H)*(2*(S-S1)**3
     2  /H**3-3*(S-S1)**2/H**2 + 1)
      ZATD(1,3) = C*(2*(2*(S-S1)/H-1)/H-1/H)*(1-(S-S1)/H)**2*(S-S1) + C*((
     1  2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)*(2*(1-(S-S1)/H)*(S-S1)
     2  /H-(1-(S-S1)/H)**2)
      ZATD(1,4) = C**2*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)*((2*(S
     1  -S1)/H-1)**2/2.0 + (2*(S-S1)/H-1)/2.0) + (2*(2*(S-S1)/H-1)/H-1/H)*(
     2  2*(2*(S-S1)/H-1)/H + 1/H)
      ZATD(1,5) = C*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)*(6*(S-S1)
     1  /H**2-6*(S-S1)**2/H**3)-C*(2*(2*(S-S1)/H-1)/H-1/H)*(3*(S-S1)**2
     2  /H**2-2*(S-S1)**3/H**3)
      ZATD(1,6) = C*(2*(2*(S-S1)/H-1)/H-1/H)*((S-S1)**2/H**2-(S-S1)/H)*(
     1  S-S1) + C*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)*(-(S-S1)**2/
     2  H**2-(2*(S-S1)/H**2-1/H)*(S-S1) + (S-S1)/H)
      ZATD(1,7) = C**2*(1-(2*(S-S1)/H-1)**2)*((2*(S-S1)/H-1)**2/2.0-(2*(
     1  S-S1)/H-1)/2.0)-4*(2*(2*(S-S1)/H-1)/H-1/H)*(2*(S-S1)/H-1)/H
      ZATD(2,1) = C*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)*(6*(S-S1)
     1  **2/H**3-6*(S-S1)/H**2)-C*(2*(2*(S-S1)/H-1)/H-1/H)*(2*(S-S1)**3
     2  /H**3-3*(S-S1)**2/H**2 + 1)
      ZATD(2,2) = C**2*(2*(S-S1)**3/H**3-3*(S-S1)**2/H**2 + 1)**2 + (6*(S-S1
     1  )**2/H**3-6*(S-S1)/H**2)**2
      ZATD(2,3) = (2*(1-(S-S1)/H)*(S-S1)/H-(1-(S-S1)/H)**2)*(6*(S-S1)**2
     1  /H**3-6*(S-S1)/H**2)-C**2*(1-(S-S1)/H)**2*(2*(S-S1)**3/H**3-3*(
     2  S-S1)**2/H**2 + 1)*(S-S1)
      ZATD(2,4) = C*((2*(S-S1)/H-1)**2/2.0 + (2*(S-S1)/H-1)/2.0)*(6*(S-S1)
     1  **2/H**3-6*(S-S1)/H**2)-C*(2*(2*(S-S1)/H-1)/H + 1/H)*(2*(S-S1)**3
     2  /H**3-3*(S-S1)**2/H**2 + 1)
      ZATD(2,5) = C**2*(3*(S-S1)**2/H**2-2*(S-S1)**3/H**3)*(2*(S-S1)**3/
     1  H**3-3*(S-S1)**2/H**2 + 1) + (6*(S-S1)/H**2-6*(S-S1)**2/H**3)*(6*(S
     2  -S1)**2/H**3-6*(S-S1)/H**2)
      ZATD(2,6) = (6*(S-S1)**2/H**3-6*(S-S1)/H**2)*(-(S-S1)**2/H**2-(2*(
     1  S-S1)/H**2-1/H)*(S-S1) + (S-S1)/H)-C**2*((S-S1)**2/H**2-(S-S1)/H)
     2  *(2*(S-S1)**3/H**3-3*(S-S1)**2/H**2 + 1)*(S-S1)
      ZATD(2,7) = 4*C*(2*(S-S1)/H-1)*(2*(S-S1)**3/H**3-3*(S-S1)**2/H**2 +
     1  1)/H + C*(1-(2*(S-S1)/H-1)**2)*(6*(S-S1)**2/H**3-6*(S-S1)/H**2)
      ZATD(3,1) = C*(2*(2*(S-S1)/H-1)/H-1/H)*(1-(S-S1)/H)**2*(S-S1) + C*((
     1  2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)*(2*(1-(S-S1)/H)*(S-S1)
     2  /H-(1-(S-S1)/H)**2)
```

```
      ZATD(3,2) = (2*(1-(S-S1)/H)*(S-S1)/H-(1-(S-S1)/H)**2)*(6*(S-S1)**2
     1  /H**3-6*(S-S1)/H**2)-C**2*(1-(S-S1)/H)**2*(2*(S-S1)**3/H**3-3*(
     2  S-S1)**2/H**2+1)*(S-S1)
      ZATD(3,3) = C**2*(1-(S-S1)/H)**4*(S-S1)**2+(2*(1-(S-S1)/H)*(S-S1)/
     1  H-(1-(S-S1)/H)**2)**2
      ZATD(3,4) = C*(2*(2*(S-S1)/H-1)/H+1/H)*(1-(S-S1)/H)**2*(S-S1)+C*((
     1  2*(S-S1)/H-1)**2/2.0+(2*(S-S1)/H-1)/2.0)*(2*(1-(S-S1)/H)*(S-S1)
     2  /H-(1-(S-S1)/H)**2)
      ZATD(3,5) = (2*(1-(S-S1)/H)*(S-S1)/H-(1-(S-S1)/H)**2)*(6*(S-S1)/H*
     1  *2-6*(S-S1)**2/H**3)-C**2*(1-(S-S1)/H)**2*(3*(S-S1)**2/H**2-2*(
     2  S-S1)**3/H**3)*(S-S1)
      ZATD(3,6) = C**2*(1-(S-S1)/H)**2*((S-S1)**2/H**2-(S-S1)/H)*(S-S1)*
     1  *2+(2*(1-(S-S1)/H)*(S-S1)/H-(1-(S-S1)/H)**2)*(-(S-S1)**2/H**2-(
     2  2*(S-S1)/H**2-1/H)*(S-S1)+(S-S1)/H)
      ZATD(3,7) = C*(1-(2*(S-S1)/H-1)**2)*(2*(1-(S-S1)/H)*(S-S1)/H-(1-(S
     1  -S1)/H)**2)-4*C*(1-(S-S1)/H)**2*(2*(S-S1)/H-1)*(S-S1)/H
      ZATD(4,1) = C**2*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)*((2*(S
     1  -S1)/H-1)**2/2.0+(2*(S-S1)/H-1)/2.0)+(2*(2*(S-S1)/H-1)/H-1/H)*(
     2  2*(2*(S-S1)/H-1)/H+1/H)
      ZATD(4,2) = C*((2*(S-S1)/H-1)**2/2.0+(2*(S-S1)/H-1)/2.0)*(6*(S-S1)
     1  **2/H**3-6*(S-S1)/H**2)-C*(2*(2*(S-S1)/H-1)/H+1/H)*(2*(S-S1)**3
     2  /H**3-3*(S-S1)**2/H**2+1)
      ZATD(4,3) = C*(2*(2*(S-S1)/H-1)/H+1/H)*(1-(S-S1)/H)**2*(S-S1)+C*((
     1  2*(S-S1)/H-1)**2/2.0+(2*(S-S1)/H-1)/2.0)*(2*(1-(S-S1)/H)*(S-S1)
     2  /H-(1-(S-S1)/H)**2)
      ZATD(4,4) = C**2*((2*(S-S1)/H-1)**2/2.0+(2*(S-S1)/H-1)/2.0)**2+(2*
     1  (2*(S-S1)/H-1)/H+1/H)**2
      ZATD(4,5) = C*((2*(S-S1)/H-1)**2/2.0+(2*(S-S1)/H-1)/2.0)*(6*(S-S1)
     1  /H**2-6*(S-S1)**2/H**3)-C*(2*(2*(S-S1)/H-1)/H+1/H)*(3*(S-S1)**2
     2  /H**2-2*(S-S1)**3/H**3)
      ZATD(4,6) = C*(2*(2*(S-S1)/H-1)/H+1/H)*((S-S1)**2/H**2-(S-S1)/H)*(
     1  S-S1)+C*((2*(S-S1)/H-1)**2/2.0+(2*(S-S1)/H-1)/2.0)*(-(S-S1)**2/
     2  H**2-(2*(S-S1)/H**2-1/H)*(S-S1)+(S-S1)/H)
      ZATD(4,7) = C**2*(1-(2*(S-S1)/H-1)**2)*((2*(S-S1)/H-1)**2/2.0+(2*(
     1  S-S1)/H-1)/2.0)-4*(2*(2*(S-S1)/H-1)/H+1/H)*(2*(S-S1)/H-1)/H
      ZATD(5,1) = C*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)*(6*(S-S1)
     1  /H**2-6*(S-S1)**2/H**3)-C*(2*(2*(S-S1)/H-1)/H-1/H)*(3*(S-S1)**2
     2  /H**2-2*(S-S1)**3/H**3)
      ZATD(5,2) = C**2*(3*(S-S1)**2/H**2-2*(S-S1)**3/H**3)*(2*(S-S1)**3/
     1  H**3-3*(S-S1)**2/H**2+1)+(6*(S-S1)/H**2-6*(S-S1)**2/H**3)*(6*(S
     2  -S1)**2/H**3-6*(S-S1)/H**2)
      ZATD(5,3) = (2*(1-(S-S1)/H)*(S-S1)/H-(1-(S-S1)/H)**2)*(6*(S-S1)/H*
     1  *2-6*(S-S1)**2/H**3)-C**2*(1-(S-S1)/H)**2*(3*(S-S1)**2/H**2-2*(
     2  S-S1)**3/H**3)*(S-S1)
      ZATD(5,4) = C*((2*(S-S1)/H-1)**2/2.0+(2*(S-S1)/H-1)/2.0)*(6*(S-S1)
     1  /H**2-6*(S-S1)**2/H**3)-C*(2*(2*(S-S1)/H-1)/H+1/H)*(3*(S-S1)**2
     2  /H**2-2*(S-S1)**3/H**3)
      ZATD(5,5) = C**2*(3*(S-S1)**2/H**2-2*(S-S1)**3/H**3)**2+(6*(S-S1)/
     1  H**2-6*(S-S1)**2/H**3)**2
      ZATD(5,6) = (6*(S-S1)/H**2-6*(S-S1)**2/H**3)*(-(S-S1)**2/H**2-(2*(
     1  S-S1)/H**2-1/H)*(S-S1)+(S-S1)/H)-C**2*((S-S1)**2/H**2-(S-S1)/H)
     2  *(3*(S-S1)**2/H**2-2*(S-S1)**3/H**3)*(S-S1)
      ZATD(5,7) = 4*C*(2*(S-S1)/H-1)*(3*(S-S1)**2/H**2-2*(S-S1)**3/H**3)
     1  /H+C*(1-(2*(S-S1)/H-1)**2)*(6*(S-S1)/H**2-6*(S-S1)**2/H**3)
      ZATD(6,1) = C*(2*(2*(S-S1)/H-1)/H-1/H)*((S-S1)**2/H**2-(S-S1)/H)*(
     1  S-S1)+C*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)*(-(S-S1)**2/
```

```
     2  H**2-(2*(S-S1)/H**2-1/H)*(S-S1) + (S-S1)/H)
      ZATD(6,2) = (6*(S-S1)**2/H**3-6*(S-S1)/H**2)*(-(S-S1)**2/H**2-(2*(
     1  S-S1)/H**2-1/H)*(S-S1) + (S-S1)/H)-C**2*((S-S1)**2/H**2-(S-S1)/H)
     2  *(2*(S-S1)**3/H**3-3*(S-S1)**2/H**2 + 1)*(S-S1)
      ZATD(6,3) = C**2*(1-(S-S1)/H)**2*((S-S1)**2/H**2-(S-S1)/H)*(S-S1)*
     1  *2 + (2*(1-(S-S1)/H)*(S-S1)/H-(1-(S-S1)/H)**2)*(-(S-S1)**2/H**2-(
     2  2*(S-S1)/H**2-1/H)*(S-S1) + (S-S1)/H)
      ZATD(6,4) = C*(2*(2*(S-S1)/H-1)/H + 1/H)*((S-S1)**2/H**2-(S-S1)/H)*(
     1  S-S1) + C*((2*(S-S1)/H-1)**2/2.0 + (2*(S-S1)/H-1)/2.0)*(-(S-S1)**2/
     2  H**2-(2*(S-S1)/H**2-1/H)*(S-S1) + (S-S1)/H)
      ZATD(6,5) = (6*(S-S1)/H**2-6*(S-S1)**2/H**3)*(-(S-S1)**2/H**2-(2*(
     1  S-S1)/H**2-1/H)*(S-S1) + (S-S1)/H)-C**2*((S-S1)**2/H**2-(S-S1)/H)
     2  *(3*(S-S1)**2/H**2-2*(S-S1)**3/H**3)*(S-S1)
      ZATD(6,6) = C**2*((S-S1)**2/H**2-(S-S1)/H)**2*(S-S1)**2 + (-(S-S1)**
     1  2/H**2-(2*(S-S1)/H**2-1/H)*(S-S1) + (S-S1)/H)**2
      ZATD(6,7) = C*(1-(2*(S-S1)/H-1)**2)*(-(S-S1)**2/H**2-(2*(S-S1)/H**
     1  2-1/H)*(S-S1) + (S-S1)/H)-4*C*(2*(S-S1)/H-1)*((S-S1)**2/H**2-(S-S
     2  1)/H)*(S-S1)/H
      ZATD(7,1) = C**2*(1-(2*(S-S1)/H-1)**2)*((2*(S-S1)/H-1)**2/2.0-(2*(
     1  S-S1)/H-1)/2.0)-4*(2*(2*(S-S1)/H-1)/H-1/H)*(2*(S-S1)/H-1)/H
      ZATD(7,2) = 4*C*(2*(S-S1)/H-1)*(2*(S-S1)**3/H**3-3*(S-S1)**2/H**2 +
     1  1)/H + C*(1-(2*(S-S1)/H-1)**2)*(6*(S-S1)**2/H**3-6*(S-S1)/H**2)
      ZATD(7,3) = C*(1-(2*(S-S1)/H-1)**2)*(2*(1-(S-S1)/H)*(S-S1)/H-(1-(S
     1  -S1)/H)**2)-4*C*(1-(S-S1)/H)**2*(2*(S-S1)/H-1)*(S-S1)/H
      ZATD(7,4) = C**2*(1-(2*(S-S1)/H-1)**2)*((2*(S-S1)/H-1)**2/2.0 + (2*(
     1  S-S1)/H-1)/2.0)-4*(2*(2*(S-S1)/H-1)/H + 1/H)*(2*(S-S1)/H-1)/H
      ZATD(7,5) = 4*C*(2*(S-S1)/H-1)*(3*(S-S1)**2/H**2-2*(S-S1)**3/H**3)
     1  /H + C*(1-(2*(S-S1)/H-1)**2)*(6*(S-S1)/H**2-6*(S-S1)**2/H**3)
      ZATD(7,6) = C*(1-(2*(S-S1)/H-1)**2)*(-(S-S1)**2/H**2-(2*(S-S1)/H**
     1  2-1/H)*(S-S1) + (S-S1)/H)-4*C*(2*(S-S1)/H-1)*((S-S1)**2/H**2-(S-S
     2  1)/H)*(S-S1)/H
      ZATD(7,7) = 16*(2*(S-S1)/H-1)**2/H**2 + C**2*(1-(2*(S-S1)/H-1)**2)**
     1  2
      RETURN
      END
C ***********************************************************************
C CODE FOR MATRIX E1
C ***********************************************************************
      SUBROUTINE MATXE1(S,S1,H,C,ZATE1)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION ZATE1(7,7)
      ZATE1(1,1) = C**2*(2*(2*(S-S1)/H-1)/H-1/H)*((2*(S-S1)/H-1)**2/2.0-
     1  (2*(S-S1)/H-1)/2.0)
      ZATE1(1,2) = C*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)*(12*(S-S
     1  1)/H**3-6/H**2)
      ZATE1(1,3) = C*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)*(4*(1-(S
     1  -S1)/H)/H-2*(S-S1)/H**2)
      ZATE1(1,4) = C**2*(2*(2*(S-S1)/H-1)/H + 1/H)*((2*(S-S1)/H-1)**2/2.0-
     1  (2*(S-S1)/H-1)/2.0)
      ZATE1(1,5) = C*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-  0)*(6/H**2-
     1  12*(S-S1)/H**3)
      ZATE1(1,6) = C*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-  0)*(2/H-6*(
     1  S-S1)/H**2)
      ZATE1(1,7) = -4*C**2*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)*(2
     1  *(S-S1)/H-1)/H
      ZATE1(2,1) = C*(2*(2*(S-S1)/H-1)/H-1/H)*(6*(S-S1)**2/H**3-6*(S-S1)
```

```
1   /H**2)
     ZATE1(2,2) = (12*(S-S1)/H**3-6/H**2)*(6*(S-S1)**2/H**3-6*(S-S1)/H*
1   *2)
     ZATE1(2,3) = (4*(1-(S-S1)/H)/H-2*(S-S1)/H**2)*(6*(S-S1)**2/H**3-6*
1   (S-S1)/H**2)
     ZATE1(2,4) = C*(2*(2*(S-S1)/H-1)/H + 1/H)*(6*(S-S1)**2/H**3-6*(S-S1)
1   /H**2)
     ZATE1(2,5) = (6/H**2-12*(S-S1)/H**3)*(6*(S-S1)**2/H**3-6*(S-S1)/H*
1   *2)
     ZATE1(2,6) = (2/H-6*(S-S1)/H**2)*(6*(S-S1)**2/H**3-6*(S-S1)/H**2)
     ZATE1(2,7) = -4*C*(2*(S-S1)/H-1)*(6*(S-S1)**2/H**3-6*(S-S1)/H**2)/
1   H
     ZATE1(3,1) = C*(2*(2*(S-S1)/H-1)/H-1/H)*(2*(1-(S-S1)/H)*(S-S1)/H-(
1   1-(S-S1)/H)**2)
     ZATE1(3,2) = (12*(S-S1)/H**3-6/H**2)*(2*(1-(S-S1)/H)*(S-S1)/H-(1-(
1   S-S1)/H)**2)
     ZATE1(3,3) = (4*(1-(S-S1)/H)/H-2*(S-S1)/H**2)*(2*(1-(S-S1)/H)*(S-S
1   1)/H-(1-(S-S1)/H)**2)
     ZATE1(3,4) = C*(2*(2*(S-S1)/H-1)/H + 1/H)*(2*(1-(S-S1)/H)*(S-S1)/H-(
1   1-(S-S1)/H)**2)
     ZATE1(3,5) = (6/H**2-12*(S-S1)/H**3)*(2*(1-(S-S1)/H)*(S-S1)/H-(1-(
1   S-S1)/H)**2)
     ZATE1(3,6) = (2/H-6*(S-S1)/H**2)*(2*(1-(S-S1)/H)*(S-S1)/H-(1-(S-S1
1   )/H)**2)
     ZATE1(3,7) = -4*C*(2*(S-S1)/H-1)*(2*(1-(S-S1)/H)*(S-S1)/H-(1-(S-S1
1   )/H)**2)/H
     ZATE1(4,1) = C**2*(2*(2*(S-S1)/H-1)/H-1/H)*((2*(S-S1)/H-1)**2/2.0+
1   (2*(S-S1)/H-1)/2.0)
     ZATE1(4,2) = C*((2*(S-S1)/H-1)**2/2.0 + (2*(S-S1)/H-1)/2.0)*(12*(S-S
1   1)/H**3-6/H**2)
     ZATE1(4,3) = C*((2*(S-S1)/H-1)**2/2.0 + (2*(S-S1)/H-1)/2.0)*(4*(1-(S
1   -S1)/H)/H-2*(S-S1)/H**2)
     ZATE1(4,4) = C**2*(2*(2*(S-S1)/H-1)/H + 1/H)*((2*(S-S1)/H-1)**2/2.0+
1   (2*(S-S1)/H-1)/2.0)
     ZATE1(4,5) = C*((2*(S-S1)/H-1)**2/2.0 + (2*(S-S1)/H-1)/2.0)*(6/H**2-
1   12*(S-S1)/H**3)
     ZATE1(4,6) = C*((2*(S-S1)/H-1)**2/2.0 + (2*(S-S1)/H-1)/2.0)*(2/H-6*(
1   S-S1)/H**2)
     ZATE1(4,7) = -4*C**2*((2*(S-S1)/H-1)**2/2.0 + (2*(S-S1)/H-1)/2.0)*(2
1   *(S-S1)/H-1)/H
     ZATE1(5,1) = C*(2*(2*(S-S1)/H-1)/H-1/H)*(6*(S-S1)/H**2-6*(S-S1)**2
1   /H**3)
     ZATE1(5,2) = (12*(S-S1)/H**3-6/H**2)*(6*(S-S1)/H**2-6*(S-S1)**2/H*
1   *3)
     ZATE1(5,3) = (4*(1-(S-S1)/H)/H-2*(S-S1)/H**2)*(6*(S-S1)/H**2-6*(S-
1   S1)**2/H**3)
     ZATE1(5,4) = C*(2*(2*(S-S1)/H-1)/H + 1/H)*(6*(S-S1)/H**2-6*(S-S1)**2
1   /H**3)
     ZATE1(5,5) = (6/H**2-12*(S-S1)/H**3)*(6*(S-S1)/H**2-6*(S-S1)**2/H*
1   *3)
     ZATE1(5,6) = (2/H-6*(S-S1)/H**2)*(6*(S-S1)/H**2-6*(S-S1)**2/H**3)
     ZATE1(5,7) = -4*C*(2*(S-S1)/H-1)*(6*(S-S1)/H**2-6*(S-S1)**2/H**3)/
1   H
     ZATE1(6,1) = C*(2*(2*(S-S1)/H-1)/H-1/H)*(-(S-S1)**2/H**2-(2*(S-S1)
1   /H**2-1/H)*(S-S1) + (S-S1)/H)
     ZATE1(6,2) = (12*(S-S1)/H**3-6/H**2)*(-(S-S1)**2/H**2-(2*(S-S1)/H*
```

```
     1   *2-1/H)*(S-S1) + (S-S1)/H)
         ZATE1(6,3) = (4*(1-(S-S1)/H)/H-2*(S-S1)/H**2)*(-(S-S1)**2/H**2-(2*
     1   (S-S1)/H**2-1/H)*(S-S1) + (S-S1)/H)
         ZATE1(6,4) = C*(2*(2*(S-S1)/H-1)/H + 1/H)*(-(S-S1)**2/H**2-(2*(S-S1)
     1   /H**2-1/H)*(S-S1) + (S-S1)/H)
         ZATE1(6,5) = (6/H**2-12*(S-S1)/H**3)*(-(S-S1)**2/H**2-(2*(S-S1)/H*
     1   *2-1/H)*(S-S1) + (S-S1)/H)
         ZATE1(6,6) = (2/H-6*(S-S1)/H**2)*(-(S-S1)**2/H**2-(2*(S-S1)/H**2-1
     1   /H)*(S-S1) + (S-S1)/H)
         ZATE1(6,7) = -4*C*(2*(S-S1)/H-1)*(-(S-S1)**2/H**2-(2*(S-S1)/H**2-1
     1   /H)*(S-S1) + (S-S1)/H)/H
         ZATE1(7,1) = C**2*(2*(2*(S-S1)/H-1)/H-1/H)*(1-(2*(S-S1)/H-1)**2)
         ZATE1(7,2) = C*(1-(2*(S-S1)/H-1)**2)*(12*(S-S1)/H**3-6/H**2)
         ZATE1(7,3) = C*(1-(2*(S-S1)/H-1)**2)*(4*(1-(S-S1)/H)/H-2*(S-S1)/H*
     1   *2)
         ZATE1(7,4) = C**2*(2*(2*(S-S1)/H-1)/H + 1/H)*(1-(2*(S-S1)/H-1)**2)
         ZATE1(7,5) = C*(1-(2*(S-S1)/H-1)**2)*(6/H**2-12*(S-S1)/H**3)
         ZATE1(7,6) = C*(1-(2*(S-S1)/H-1)**2)*(2/H-6*(S-S1)/H**2)
         ZATE1(7,7) = -4*C**2*(1-(2*(S-S1)/H-1)**2)*(2*(S-S1)/H-1)/H
         RETURN
         END
C ***************************************************************************
C CODE FOR MATRIX E2
C ***************************************************************************
         SUBROUTINE MATXE2(S,S1,H,C,ZATE2)
         IMPLICIT DOUBLE PRECISION (A-H,O-Z)
         DIMENSION ZATE2(7,7)
         ZATE2(1,1) = C**2*(2*(2*(S-S1)/H-1)/H-1/H)*((2*(S-S1)/H-1)**2/2.0-
     1   (2*(S-S1)/H-1)/2.0)
         ZATE2(1,2) = C*(2*(2*(S-S1)/H-1)/H-1/H)*(6*(S-S1)**2/H**3-6*(S-S1)
     1   /H**2)
         ZATE2(1,3) = C*(2*(2*(S-S1)/H-1)/H-1/H)*(2*(1-(S-S1)/H)*(S-S1)/H-(
     1   1-(S-S1)/H)**2)
         ZATE2(1,4) = C**2*(2*(2*(S-S1)/H-1)/H-1/H)*((2*(S-S1)/H-1)**2/2.0 +
     1   (2*(S-S1)/H-1)/2.0)
         ZATE2(1,5) = C*(2*(2*(S-S1)/H-1)/H-1/H)*(6*(S-S1)/H**2-6*(S-S1)**2
     1   /H**3)
         ZATE2(1,6) = C*(2*(2*(S-S1)/H-1)/H-1/H)*(-(S-S1)**2/H**2-(2*(S-S1)
     1   /H**2-1/H)*(S-S1) + (S-S1)/H)
         ZATE2(1,7) = C**2*(2*(2*(S-S1)/H-1)/H-1/H)*(1-(2*(S-S1)/H-1)**2)
         ZATE2(2,1) = C*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)*(12*(S-S
     1   1)/H**3-6/H**2)
         ZATE2(2,2) = (12*(S-S1)/H**3-6/H**2)*(6*(S-S1)**2/H**3-6*(S-S1)/H*
     1   *2)
         ZATE2(2,3) = (12*(S-S1)/H**3-6/H**2)*(2*(1-(S-S1)/H)*(S-S1)/H-(1-(
     1   S-S1)/H)**2)
         ZATE2(2,4) = C*((2*(S-S1)/H-1)**2/2.0 + (2*(S-S1)/H-1)/2.0)*(12*(S-S
     1   1)/H**3-6/H**2)
         ZATE2(2,5) = (12*(S-S1)/H**3-6/H**2)*(6*(S-S1)/H**2-6*(S-S1)**2/H*
     1   *3)
         ZATE2(2,6) = (12*(S-S1)/H**3-6/H**2)*(-(S-S1)**2/H**2-(2*(S-S1)/H*
     1   *2-1/H)*(S-S1) + (S-S1)/H)
         ZATE2(2,7) = C*(1-(2*(S-S1)/H-1)**2)*(12*(S-S1)/H**3-6/H**2)
         ZATE2(3,1) = C*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)*(4*(1-(S
     1   -S1)/H)/H-2*(S-S1)/H**2)
         ZATE2(3,2) = (4*(1-(S-S1)/H)/H-2*(S-S1)/H**2)*(6*(S-S1)**2/H**3-6*
```

```
1   (S-S1)/H**2)
 ZATE2(3,3) = (4*(1-(S-S1)/H)/H-2*(S-S1)/H**2)*(2*(1-(S-S1)/H)*(S-S
1   1)/H-(1-(S-S1)/H)**2)
 ZATE2(3,4) = C*((2*(S-S1)/H-1)**2/2.0+(2*(S-S1)/H-1)/2.0)*(4*(1-(S
1   -S1)/H)/H-2*(S-S1)/H**2)
 ZATE2(3,5) = (4*(1-(S-S1)/H)/H-2*(S-S1)/H**2)*(6*(S-S1)/H**2-6*(S-
1   S1)**2/H**3)
 ZATE2(3,6) = (4*(1-(S-S1)/H)/H-2*(S-S1)/H**2)*(-(S-S1)**2/H**2-(2*
1   (S-S1)/H**2-1/H)*(S-S1)+(S-S1)/H)
 ZATE2(3,7) = C*(1-(2*(S-S1)/H-1)**2)*(4*(1-(S-S1)/H)/H-2*(S-S1)/H*
1   *2)
 ZATE2(4,1) = C**2*(2*(2*(S-S1)/H-1)/H+1/H)*((2*(S-S1)/H-1)**2/2.0-
1   (2*(S-S1)/H-1)/2.0)
 ZATE2(4,2) = C*(2*(2*(S-S1)/H-1)/H+1/H)*(6*(S-S1)**2/H**3-6*(S-S1)
1   /H**2)
 ZATE2(4,3) = C*(2*(2*(S-S1)/H-1)/H+1/H)*(2*(1-(S-S1)/H)*(S-S1)/H-(
1   1-(S-S1)/H)**2)
 ZATE2(4,4) = C**2*(2*(2*(S-S1)/H-1)/H+1/H)*((2*(S-S1)/H-1)**2/2.0+
1   (2*(S-S1)/H-1)/2.0)
 ZATE2(4,5) = C*(2*(2*(S-S1)/H-1)/H+1/H)*(6*(S-S1)/H**2-6*(S-S1)**2
1   /H**3)
 ZATE2(4,6) = C*(2*(2*(S-S1)/H-1)/H+1/H)*(-(S-S1)**2/H**2-(2*(S-S1)
1   /H**2-1/H)*(S-S1)+(S-S1)/H)
 ZATE2(4,7) = C**2*(2*(2*(S-S1)/H-1)/H+1/H)*(1-(2*(S-S1)/H-1)**2)
 ZATE2(5,1) = C*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)*(6/H**2-
1   12*(S-S1)/H**3)
 ZATE2(5,2) = (6/H**2-12*(S-S1)/H**3)*(6*(S-S1)**2/H**3-6*(S-S1)/H*
1   *2)
 ZATE2(5,3) = (6/H**2-12*(S-S1)/H**3)*(2*(1-(S-S1)/H)*(S-S1)/H-(1-(
1   S-S1)/H)**2)
 ZATE2(5,4) = C*((2*(S-S1)/H-1)**2/2.0+(2*(S-S1)/H-1)/2.0)*(6/H**2-
1   12*(S-S1)/H**3)
 ZATE2(5,5) = (6/H**2-12*(S-S1)/H**3)*(6*(S-S1)/H**2-6*(S-S1)**2/H*
1   *3)
 ZATE2(5,6) = (6/H**2-12*(S-S1)/H**3)*(-(S-S1)**2/H**2-(2*(S-S1)/H*
1   *2-1/H)*(S-S1)+(S-S1)/H)
 ZATE2(5,7) = C*(1-(2*(S-S1)/H-1)**2)*(6/H**2-12*(S-S1)/H**3)
 ZATE2(6,1) = C*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)*(2/H-6*(
1   S-S1)/H**2)
 ZATE2(6,2) = (2/H-6*(S-S1)/H**2)*(6*(S-S1)**2/H**3-6*(S-S1)/H**2)
 ZATE2(6,3) = (2/H-6*(S-S1)/H**2)*(2*(1-(S-S1)/H)*(S-S1)/H-(1-(S-S1
1   )/H)**2)
 ZATE2(6,4) = C*((2*(S-S1)/H-1)**2/2.0+(2*(S-S1)/H-1)/2.0)*(2/H-6*(
1   S-S1)/H**2)
 ZATE2(6,5) = (2/H-6*(S-S1)/H**2)*(6*(S-S1)/H**2-6*(S-S1)**2/H**3)
 ZATE2(6,6) = (2/H-6*(S-S1)/H**2)*(-(S-S1)**2/H**2-(2*(S-S1)/H**2-1
1   /H)*(S-S1)+(S-S1)/H)
 ZATE2(6,7) = C*(1-(2*(S-S1)/H-1)**2)*(2/H-6*(S-S1)/H**2)
 ZATE2(7,1) = -4*C**2*((2*(S-S1)/H-1)**2/2.0-(2*(S-S1)/H-1)/2.0)*(2
1   *(S-S1)/H-1)/H
 ZATE2(7,2) = -4*C*(2*(S-S1)/H-1)*(6*(S-S1)**2/H**3-6*(S-S1)/H**2)/
1   H
 ZATE2(7,3) = -4*C*(2*(S-S1)/H-1)*(2*(1-(S-S1)/H)*(S-S1)/H-(1-(S-S1
1   )/H)**2)/H
 ZATE2(7,4) = -4*C**2*((2*(S-S1)/H-1)**2/2.0+(2*(S-S1)/H-1)/2.0)*(2
1   *(S-S1)/H-1)/H
```

```
      ZATE2(7,5) = -4*C*(2*(S-S1)/H-1)*(6*(S-S1)/H**2-6*(S-S1)**2/H**3)/
     1  H
      ZATE2(7,6) = -4*C*(2*(S-S1)/H-1)*(-(S-S1)**2/H**2-(2*(S-S1)/H**2-1
     1  /H)*(S-S1)+(S-S1)/H)/H
      ZATE2(7,7) = -4*C**2*(1-(2*(S-S1)/H-1)**2)*(2*(S-S1)/H-1)/H
      RETURN
      END
C ***********************************************************************
C PROGRAM TO ADD TWO COLUMN VECTORS WITH A SCALE FACTOR FOR EACH
C ***********************************************************************
      SUBROUTINE CADD(FACA,A,FACB,B,C,N1)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C     ADDS MATRIX FACA*A TO FACB*B AND CALLS IT C
      DIMENSION A(N1),B(N1),C(N1)
      DO 100 I=1,N1
  100 C(I)=FACA*A(I) + FACB*B(I)
      RETURN
      END
C ***********************************************************************
C PROGRAM TO ADD TWO N1XN2 MATRICES WITH SCALE FACTORS FOR EACH
C ***********************************************************************
      SUBROUTINE MATADD(FACA,A,FACB,B,C,N1,N2)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C     ADDS MATRIX FACA*A TO FACB*B AND CALLS IT C
      DIMENSION A(N1,N2),B(N1,N2),C(N1,N2)
      DO 100 I=1,N1
      DO 100 J=1,N2
  100 C(I,J)=FACA*A(I,J) + FACB*B(I,J)
      RETURN
      END
C ***********************************************************************
C PROGRAM TO MULTIPLY TWO NXN MATRICES
C ***********************************************************************
      SUBROUTINE MNMULT(A,B,C,N)
C     YIELDS A X B = C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION A(N,N),B(N,N),C(N,N)
      DO 100 I=1,N
      DO 100 J=1,N
      C(I,J)=0.
      DO 100 K=1,N
C     WRITE(9,1000)I,J,K,A(I,K),B(K,J),C(I,J)
  100 C(I,J)=C(I,J) + A(I,K)*B(K,J)
      RETURN
      END
C ***********************************************************************
C PROGRAM TO MULTIPLY A ROW BY A MATRIX  I.E. {R}TRAN*[M]                *
C ***********************************************************************
      SUBROUTINE CMMULT(A,B,C,N1,N2)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION A(N1),B(N1,N2),C(N2)
      DO 10 I=1,N2
   10   C(I)=0.0D0
      DO 20 I=1,N2
        DO 20 J=1,N1
   20   C(I)=A(J)*B(J,I) + C(I)
```

```
      RETURN
      END
C ***************************************************************************
C PROGRAM TO MULTIPLY A MATRIX BY A COLUMN
C ***************************************************************************
      SUBROUTINE MCMULT(A,B,C,N1,N2)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION A(N1,N2),B(N2),C(N1)
      DO 10 I = 1,N1
   10   C(I) = 0.0D0
      DO 20 I = 1,N1
        DO 20 J = 1,N2
   20   C(I) = A(I,J)*B(J) + C(I)
      RETURN
      END
C ***************************************************************************
C **  PROGRAM TO MULTIPLY TWO COLUMNS
C ***************************************************************************
      SUBROUTINE COLMULT(A,B,C,N1)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION A(N1),B(N1)
        C = 0.0D0
      DO 20 I = 1,N1
   20   C = A(I)*B(I) + C
      RETURN
      END
C ***************************************************************************
C PROGRAM TO MULTIPLY A SCALAR BY A COLUMN OR ROW MATRIX
C ***************************************************************************
      SUBROUTINE SMULT(S,A,C,N1)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION A(N1),C(N1)
      DO 20 I = 1,N1
   20   C(I) = S*A(I)
      RETURN
      END
C***************************************************************************
C SUBPROGRAM TO MULTIPLY A  N1XN2 BY A N2XN3
C***************************************************************************
      SUBROUTINE MATMULT(A,B,C,N1,N2,N3)
C     YIELDS A X B = C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION A(N1,N2),B(N2,N3),C(N1,N3)
      DO 100 I = 1,N1
      DO 100 J = 1,N3
      C(I,J) = 0.
      DO 100 K = 1,N2
  100 C(I,J) = C(I,J) + A(I,K)*B(K,J)
      RETURN
      END
C***************************************************************************
C SUBPROGRAM TO MULTIPLY THE TRANS OF AN N2XN1 BY A N2XN3 MATRIX
C***************************************************************************
      SUBROUTINE MATTMULT(A,B,C,N1,N2,N3)
C     YIELDS A(TRANS) X B = C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
```

```
      DIMENSION A(N2,N1),B(N2,N3),C(N1,N3)
      DO 100 I = 1,N1
      DO 100 J = 1,N3
      C(I,J) = 0.
      DO 100 K = 1,N2
  100 C(I,J) = C(I,J) + A(K,I)*B(K,J)
      RETURN
      END
C*********************************************************************
C SUBPROGRAM TO ADD TERMS DUE TO THE SPRING TO THE STIFFNESS MATRIX
C*********************************************************************
      SUBROUTINE SPRING(SK,SL0,NRN,GSTIF,GR,U)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION GSTIF(NRN,NRN),GR(NRN),U(NRN)
      MN = (NRN-1)/2
      MN1 = MN + 1
      UM = U(MN)
      WM = U(MN1)
      SL = DSQRT((SL0-WM)*(SL0-WM) + UM*UM)
      GSTIF(MN,MN) = GSTIF(MN,MN) + SK/SL*(SL0*UM*UM/(SL*SL) + SL-SL0)
      GSTIF(MN,MN1) = GSTIF(MN,MN1)-SK/(SL*SL)*(SL0*UM*(SL0-WM)/SL)
      GSTIF(MN1,MN) = GSTIF(MN1,MN)-SK/(SL*SL*SL)*SL0*UM*(SL0-WM)
      GSTIF(MN1,MN1) = GSTIF(MN1,MN1) + SK/SL*(SL-SL0 + SL0*(SL0-WM)*(SL0-WM)/
     1            (SL*SL))
      GR(MN) = GR(MN) + SK*(SL-SL0)*UM/SL
      GR(MN1) = GR(MN1)-SK*(SL-SL0)*(SL0-WM)/SL
      RETURN
      END
```